Trabajo Fin de Grado

Grado en Ingeniería de las Tecnologías de Telecomunicación

Aplicación Android de soporte a un juego de mesa

Autor: Marcos Girón Fernández

Tutor: Isabel Roman Martinez

Dpto. de Ingeniería Telemática

Sevilla, 2018

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Aplicación Android de soporte a un juego de mesa

Autor:
Marcos Girón Fernández

Tutor:
Isabel Roman Martinez
Profesor titular

Dpto. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2018

Autor: Marcos Girón Fernández

Tutor: Isabel Roman Martinez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

A mi familia

A mis maestros

A mis amigos

Resumen

El avance imparable de los smartphones y la existencia de tiendas virtuales de aplicaciones móviles ha producido un cambio en el mercado, obligando a muchas empresas orientar sus productos al sector de la movilidad. Según un informe de *ditrendia* (digital marketing trends) de 2017, un 66% de la población mundial ya cuenta con móvil, y éste es el dispositivo más utilizado para acceder a internet en España, usado ya por el 94,6% de los españoles.

En este proyecto hemos planificado, implementado y desplegado una aplicación Android distribuida que complementa un juego de mesa de temática financiera. Este desarrollo se ha realizado a petición de un cliente, propietario del juego y creador de las reglas. Por lo tanto, la comunicación con él ha sido crucial en el análisis de requisitos y en el desarrollo de las pruebas.

La implementación se ha realizado con el IDE Android Studio debido a las numerosas opciones avanzadas que podemos encontrar en este entorno de desarrollo, y las facilidades que ofrece.

Abstract

The unstoppable advance of smartphones and the virtual presence of mobile application stores has produced a change in the market, forcing many companies to target their products to the mobility sector. According to a report of ditrendia (digital marketing trends) of 2017, 66% of the world's population already has a mobile phone, and this is the most used device to access the internet in Spain, already used by 94.6% of Spanish people.

In this project we have planned, implemented and deployed a distributed Android application that complements a board game with a financial theme. This development was made at the request of a client, owner of the game and creator of the rules. Therefore, communication with him has been crucial in the analysis of requirements and in the development of the tests.

The implementation has been made with the IDE Android Studio due to the many advanced options that we can find in this development environment, and the facilities it offers.

ÍNDICE

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

Vivimos en una sociedad que tiende a digitalizar gran parte de las tareas que antiguamente se realizaban manualmente o haciendo uso de una serie de componentes físicos. La llegada de los smartphones, y consecuentemente de las aplicaciones móviles, ha provocado que las personas utilicen estos dispositivos como una herramienta para realizar gran variedad de actividades que se han visto simplificadas por una pulsación en un botón.

Esta evolución ha llegado a los juegos de mesa, es más, en la gran mayoría de las situaciones los ha sustituido. Son pocos aquellos que han conseguido mantener la esencia, que es estar reunidos con los amigos y compartir una divertida partida juntos, dando lugar al fortalecimiento de las relaciones entre ellos. Pero no podemos ignorar las ventajas que ofrecen las aplicaciones móviles, como es la automatización de procesos, simplificación de tareas y la posibilidad de conectar personas ubicadas en distintos lugares.

Ésta es la reflexión que ha motivado la creación de este proyecto. La integración de una aplicación móvil en un juego de mesa tradicional ya creado, con nombre Financial Game, aprovechando los beneficios que ofrecen ambos elementos.

Además, siendo consciente de la cada vez más popularidad del sistema operativo Android y sus aplicaciones, he querido realizar un trabajo de este tema para ampliar mis conocimientos sobre la materia, y vivir en persona el proceso para el desarrollo de una aplicación móvil.

1.1 Objetivos

El objetivo principal de este proyecto es crear una aplicación Android nativa de soporte a un juego de mesa de temática financiera, y el resto de los componentes necesarios para ofrecer las funciones online. Los jugadores podrán utilizar el programa para llevar el ritmo de la partida, ver estadísticas de las puntuaciones de cada jugador, participar en la comunidad online de Financial Game, participar en partidas online, y visualizar en formato digital las reglas del juego.

Para ello, se deberá realizar las distintas actividades propias de un proceso de desarrollo software. En los siguientes capítulos del documento se tratarán cada una de estas fases.

1.2 Descripción del documento

En el segundo capítulo de este documento, mostraremos una visión general del proyecto Android y de la aplicación desarrollada. También enumeraremos las tecnologías utilizadas que nos han permitido obtener el programa deseado.

En el tercer capítulo se desarrollará la especificación de requisitos, que son un conjunto de características deseables fruto de la indagación sobre el problema y negociación con el cliente.

En el cuarto capítulo partimos de la especificación de requisitos para proceder al diseño detallado de los distintos subsistemas que componen el proyecto.

En el quinto capítulo se presentará el plan de pruebas seguido para garantizar que la aplicación surgida cumpla con los requisitos expuestos en el capítulo tercero.

Finalmente, en el sexto capítulo se extraerán una serie de conclusiones del trabajo realizado, se listarán una serie de posibles mejoras, y se indicarán los pasos a seguir para completar aquello que no hemos podido tratar.

2 ENTORNO DE TRABAJO Y TECNOLOGÍAS

A Continuación, se lista el software utilizado para el desarrollo del proyecto, además de las tecnologías elegidas para alcanzar nuestro objetivo.

2.1 Android Studio

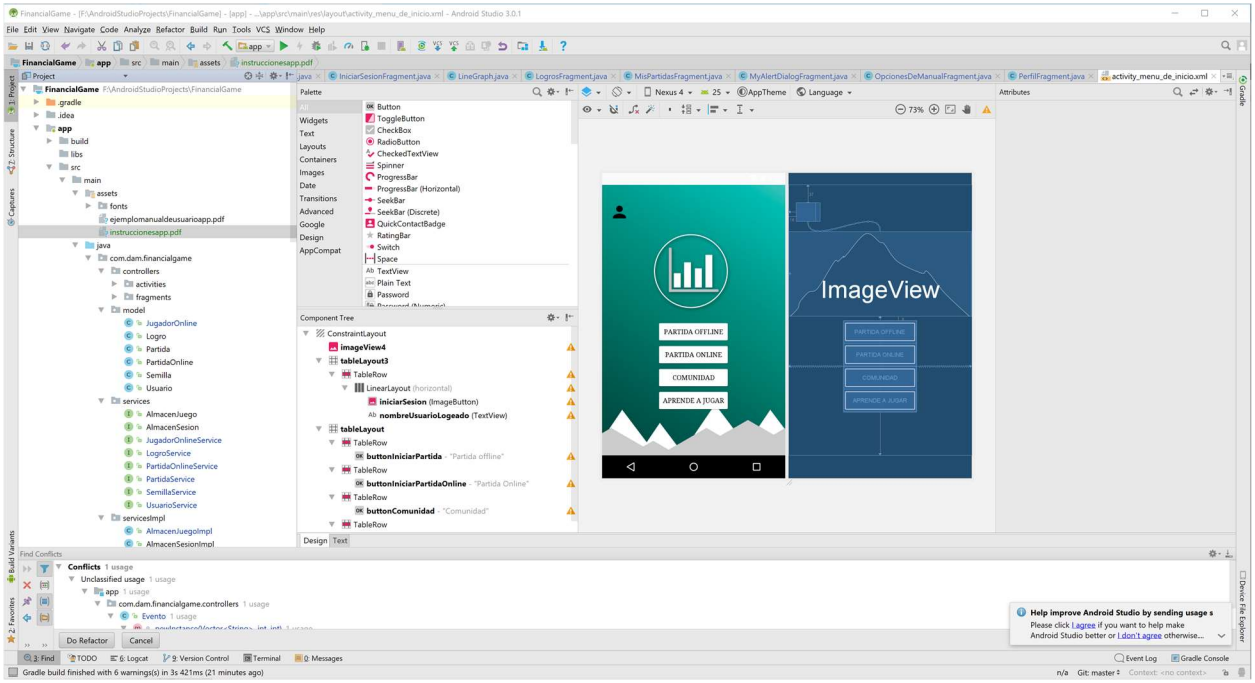


Figura: 1. Android Studio

Android Studio es el entorno de desarrollo oficial para la plataforma Android, disponible de forma gratuita y con soporte para una gran variedad de sistemas operativos. El lenguaje de programación es Java, y utiliza xml para la organización de componentes en las interfaces de usuario. Implementa Gradle para la estructuración de los proyectos, facilitando la compilación y la integración de librerías externas. Además, disponemos de una gran variedad de herramientas esenciales para la depuración de la aplicación desarrollada, como el modo debug (para depurar paso a paso) y el AVD (Administrador de Dispositivos Virtuales). Por último, este programa cuenta con la posibilidad de integrar el proyecto con repositorios de control de versiones, en nuestro caso con GitHub. Este IDE ha sido el elegido para el desarrollo de la aplicación móvil.

En la librería Java especial que ofrece esta herramienta para el desarrollo de aplicaciones móviles, debemos destacar las siguientes clases:

- Activity: Clase que representa una pantalla de la interfaz, gestiona eventos producidos por la interacción del usuario con la pantalla y muestra la información.
- Fragment: Clase que representa una parte de la interfaz gestionada por una Activity.
- AsyncTask: Clase que automatiza el uso de hilos de ejecución, procesándolos en background y devolviendo el resultado.
- Application: Clase base de la aplicación Android, y la primera en instanciarse al ejecutarse el programa.

Por último, debemos indicar que la interfaz de usuario se codifica por medio de layouts, que son archivos xml.

2.2 Control de versiones

A continuación, se muestran dos imágenes que representan los repositorios creados en la comunidad de control de versiones GitHub específicamente para el proyecto. También se indica la url debajo de cada una. Aunque ha sido un proyecto llevado a cabo por una persona, el uso del control de versiones nos ha permitido continuar el trabajo en distintos equipos y llevar un mejor control de las versiones del programa.

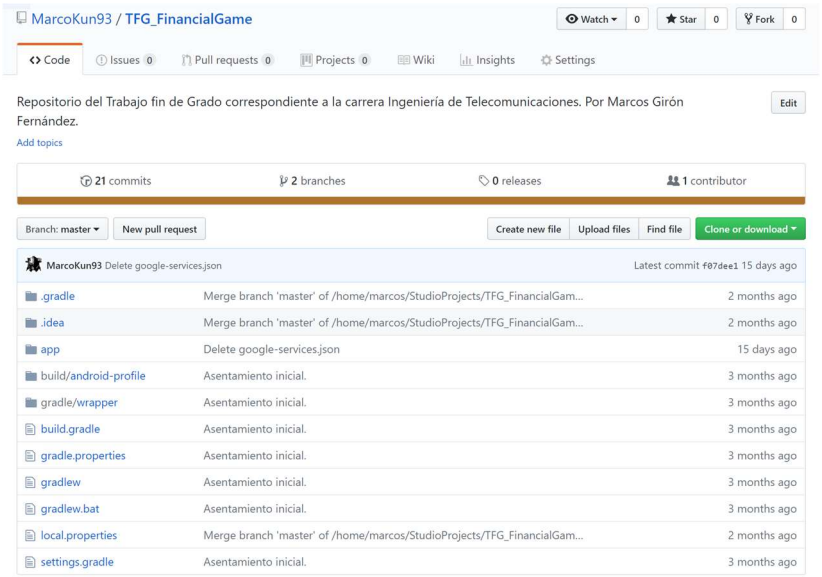


Figura: 2. https://github.com/MarcoKun93/TFG_FinancialGame

Esta figura corresponde al repositorio creado para el proyecto Android.

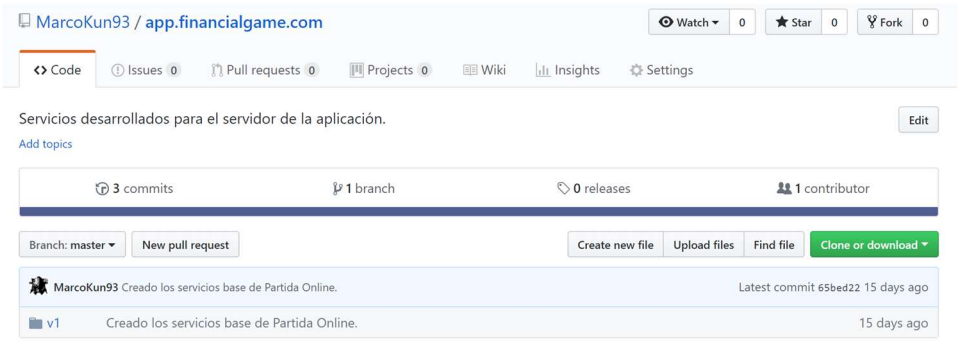


Figura: 3. https://github.com/MarcoKun93/app.financialgame.com

Esta figura corresponde al proyecto php creado para el desarrollo de los servicios web.

2.3 PhpStorm

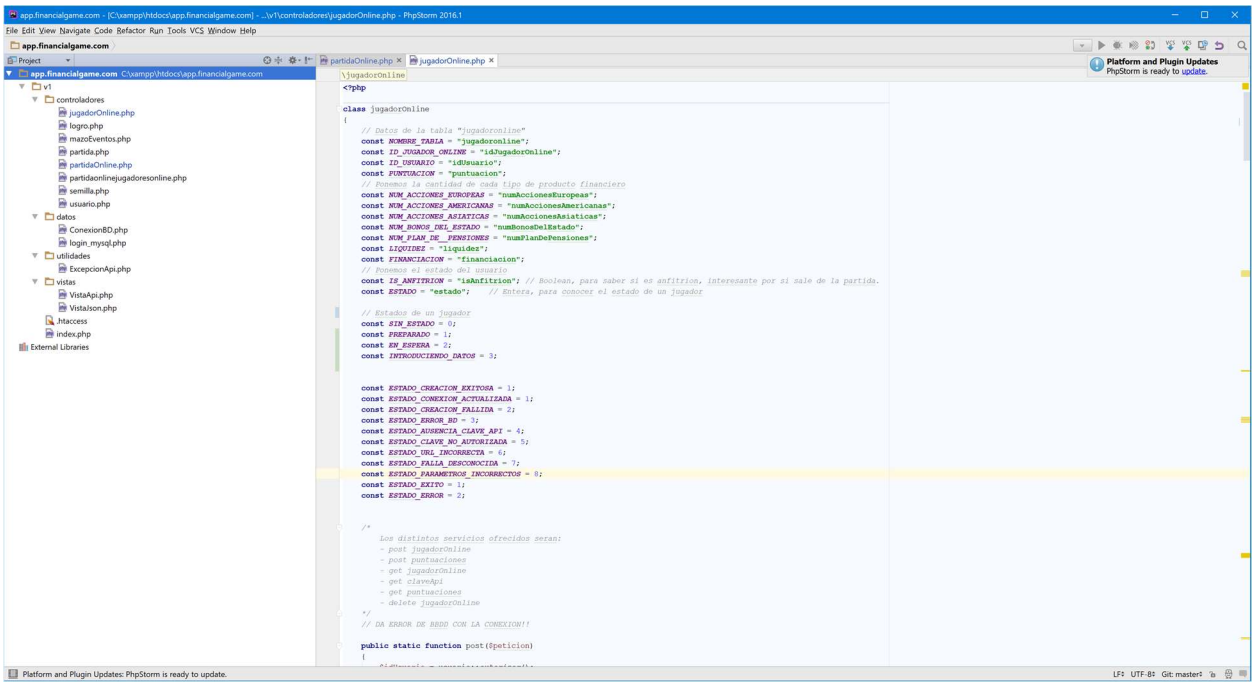


Figura: 4. PhpStorm

PhpStorm es un entorno de desarrollo integrado de carácter comercial para PHP. Ofrece un editor para dicho lenguaje y también HTML y JavaScript. Se ha utilizado para el desarrollo de los servicios web en PHP.

2.4 JSON


```
{
   "estado": 1,
  -"datos": [Array[1]
     -0:  {
          "idLogro": "2",
          "nombre": "Millonario",
          "descripcion": "El ganador ha obtenido una puntuación superior a 1.000.000",
          "tipo": "Plata",
          "variableafectada": "puntuacionganador",
          "valor": "1000000"
      }
   ],
}
```


Figura: 5. Ejemplo mensaje con estructura JSON

JSON es un formato de texto ligero para el intercambio de datos. Originalmente estaba pensado para la sintaxis de objetos de JavaScript, pero debido a su popularidad, se considera un formato de lenguaje independiente. Principalmente se ha utilizado para estructurar la información enviada en los mensajes HTTP entre cliente y servidor. Gracias a las facilidades ofrecidas por este lenguaje a la hora de decodificar la información, se han podido procesar los datos según las necesidades del proyecto. La imagen superior es un ejemplo de la información recibida del servidor, ante la petición de un servicio web, en formato JSON.

2.5 REST web services

Los servicios web son imprescindibles para cualquier desarrollo de sistemas distribuidos. Existen sitios web que ofrecen un API de métodos para ser invocados y consumirlos. Sin embargo, en nuestro caso, debemos crear ese conjunto de tareas desde cero.

Existen dos enfoques distintos para los servicios web, SOAP orientado a métodos y REST orientado a recursos. Se ha elegido la segunda opción debido a la mejor integración con las aplicaciones Android.

REST está orientado a recursos y estandariza de forma amigable su transferencia entre aplicaciones. Un recurso es todo aquello relacionado con datos los cuales estamos interesados en comunicar (en nuestro caso, registros de las tablas de la BBDD). Utiliza una estructura de urls para representar los recursos, y para poder acceder a ellos hace uso de los comandos HTTP GET, POST, PUT, y DELETE.

A continuación, se muestra una captura de las urls de los servicios web implementados, y que utilizamos en el cliente para acceder a los recursos.

```
<resources>
    <string name="app_name">Financial Game</string>
    <string name="url_registrar_usuario">http://10.0.2.2:80/app.financialgame.com/v1/usuario/registro</string>
    <string name="url_logear_usuario">http://10.0.2.2:80/app.financialgame.com/v1/usuario/login</string>
    <string name="url_cambiar_contrasenia">http://10.0.2.2:80/app.financialgame.com/v1/usuario/cambiarContrasenia</string>
    <string name="url_subir_Partida">http://10.0.2.2:80/app.financialgame.com/v1/partida/subirPartida</string>
    <string name="url_obtener_partidas_subidas">http://10.0.2.2:80/app.financialgame.com/v1/partida/partidasSubidas</string>
    <string name="url_ranking">http://10.0.2.2:80/app.financialgame.com/v1/partida/ranking</string>
    <string name="url_partida">http://10.0.2.2:80/app.financialgame.com/v1/partida</string>
    <string name="url_eliminar_partida">http://10.0.2.2:80/app.financialgame.com/v1/partida</string>
    <string name="url_actualizar_conexion">http://10.0.2.2:80/app.financialgame.com/v1/usuario/actualizarConexion</string>
    <string name="url_obtener_logros">http://10.0.2.2:80/app.financialgame.com/v1/logro/obtenerLogros</string>
    <string name="url_logro_primera_partida">http://10.0.2.2:80/app.financialgame.com/v1/logro/logroPrimeraPartida</string>
    <string name="url_logro_millonario">http://10.0.2.2:80/app.financialgame.com/v1/logro/logroMillonario</string>
    <string name="url_logro_partida_larga">http://10.0.2.2:80/app.financialgame.com/v1/logro/logroPartidaLarga</string>
    <string name="url_semilla">http://10.0.2.2:80/app.financialgame.com/v1/semilla/obtenerSemillas</string>

    <string name="pdf_instrucciones">instruccionesapp.pdf</string>
    <string name="pdf_manualDeUsuario">manualdeusuarioapp.pdf</string>

    // URL para servicios partida online
    <string name="url_get_partidas_online">http://10.0.2.2:80/app.financialgame.com/v1/partidaonline/partidasOnline</string>
    <string name="url_crear_partida_online">http://10.0.2.2:80/app.financialgame.com/v1/partidaonline/crearPartidaOnline</string>
    <string name="url_crear_jugador_online">http://10.0.2.2:80/app.financialgame.com/v1/jugadoronline/jugadorOnline</string>
    <string name="url_cambiar_estado_partida">http://10.0.2.2:80/app.financialgame.com/v1/partidaonline/postEstadoPartidaOnline</string>
    <string name="url_obtener_estado_partida">http://10.0.2.2:80/app.financialgame.com/v1/partidaonline/getEstadoPartidaOnline</string>
    <string name="url_obtener_jugadores_unidos">http://10.0.2.2:80/app.financialgame.com/v1/jugadoronline/jugadoresOnline</string>
    <string name="url_obtener_ids_online">http://10.0.2.2:80/app.financialgame.com/v1/jugadoronline/idsOnline</string>
    <string name="url_cambiar_estado_jugador">http://10.0.2.2:80/app.financialgame.com/v1/jugadoronline/postEstadoJugadorOnline</string>
    <string name="url_cambiar_numero_jugadores">http://10.0.2.2:80/app.financialgame.com/v1/partidaonline/postnumeroJugadores</string>
    <string name="url_borrar_jugador_online">http://10.0.2.2:80/app.financialgame.com/v1/jugadoronline</string> //Delete
    <string name="url_borrar_partida_online">http://10.0.2.2:80/app.financialgame.com/v1/partidaonline</string> //Delete

    // Strings de los distintos estados de una partida online
    <string name="estado_partida_online_creada">0</string>
    <string name="estado_partida_online_en_proceso">1</string>
    <string name="estado_partida_online_en_espera">2</string>
    <string name="estado_partida_online_finalizada">3</string>

    // Strings de los distintos estados de un jugador online
    <string name="estado_jugador_creado">0</string>
    <string name="estado_jugador_preparado">1</string>
    <string name="estado_jugador_en_espera">2</string>
    <string name="estado_jugador_introduciendo_datos">3</string>
```

Figura: 6. URLs de los servicios web

2.6 Volley

```
package com.dam.financialgame.threads;

import android.app.Application;

import com.android.volley.RequestQueue;
import com.android.volley.toolbox.Volley;

// La clase hará uso de un singleton para no tener que instanciarla cada vez que hagamos una petición, además
// así sólo tenemos una cola de peticiones por aplicación. Heredamos de Application con este objetivo.
public class    VolleyApplication extends Application {
    private static VolleyApplication volleyApplication;
    private RequestQueue requestQueue; // Objeto que procesa las peticiones.

    @Override
    public void onCreate() {
        super.onCreate();
        requestQueue = Volley.newRequestQueue( context: this);
        volleyApplication = this;
    }

    public synchronized static VolleyApplication getInstance() { return volleyApplication; }

    public RequestQueue getRequestQueue() { return requestQueue; }
}
```

Figura: 7. Implementación de Volley en el proyecto de Android Studio

Volley es una librería de Google que no viene integrada en el SDK de Android Studio. Es una herramienta

ideal para simplificar el proceso de conexión y descarga de información del servidor. Las llamadas a la red del cliente mediante Volley son asíncronas por lo que no tendremos que gestionar las tareas en segundo plano, ya que lo hace por nosotros. Las peticiones se gestionan por prioridades, y también tiene una caché que mejora la eficiencia del software. Existen varios tipos de peticiones, la utilizada en este proyecto en la JsonObjectRequest, que obtiene una respuesta de tipo JSONObject a partir de un recurso con este formato. Las peticiones tendrán un listener para capturar la respuesta recibida del servidor, implementando el patrón Callback.

En la imagen superior se muestra la clase *VolleyApplication* que es la encargada de gestionar las peticiones en el proyecto Android. Como es recomendable tener sólo una cola de peticiones por aplicación, se aplica el patrón Singleton.

2.7 Librerías externas utilizadas

```
repositories {
    maven { url "https://jitpack.io" }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.github.PhilJay:MPAndroidChart:v3.0.2'
    compile 'com.android.support:appcompat-v7:25.3.1'
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
    compile 'com.android.volley:volley:1.0.0'
    compile 'com.github.travijuu:numberpicker:1.0.7'
    compile 'com.github.barteksc:android-pdf-viewer:2.8.2'
    compile 'com.google.firebase:firebase-core:15.0.2'
    testCompile 'junit:junit:4.12'
}
```

Figura: 8. Dependencias de las librerías externas en Android Studio

En esta imagen se muestra la integración de las librerías externas en nuestro proyecto Android. Para ello, se debe añadir la dependencia correspondiente en el archivo *build.gradle* para que Gradle nos permita acceder a los distintos recursos externos. A continuación, se listan las librerías integradas y su uso:

- Librería Volley (com.android.volley:volley:1.0.0): Necesaria para integrar Volley en el proyecto Android.
- Librería MPAndroidChart (com.github.PhilJay:MPAndroidChart:v3.0.2): Conjunto de gráficas avanzadas diseñadas para Android. Utilizada para la representación de las puntuaciones de la partida de forma gráfica y clara.
- Librería NumberPicker (com.github.travijuu:numberpicker:1.0.7): Conjunto de elementos personalizables que hacen la misma función que un numberPicker normal de los layout de Android Studio. Utilizado para ofrecer una alternativa al elemento básico ofrecido, poco intuitivo y tosco en la interfaz de usuario, en la pantalla correspondiente a la introducción de puntuaciones.
- Librería AndroidPdfViewer (com.github.barteksc:android-pdf-viewer:2.8.2): Visualizador avanzado de archivos pdf en Android. Permite al usurio navegar cómodamente por un documento y hacer zoom. Utilizado para la visualización de las instrucciones y el manual de la aplicación.

2.8 Tokens

Para una aplicación con funcionalidades online, y que ofrecen servicios a un usuario, una de las más importantes cuestiones es cómo mantener la conexión cliente-servidor, almacenando de forma segura el identificador del usuario. Existen varias alternativas como el uso de cookies o dejarle al servidor el trabajo del mantenimiento de sesión. Sin embargo, en una aplicación Android que siempre busca la simplicidad, haremos

uso de la autenticación sin estado con Tokens. Además, este método se integra a la perfección con los servicios web REST.

En nuestro caso, el usuario se identificará haciendo uso de una clave única para cada usuario (la claveAPI) que enviará el cliente en la cabecera del mensaje HTTP, tomando el rol de un Token. Esta nueva clave la obtendrá el usuario cuando haya iniciado sesión correctamente. No es más que una firma cifrada que permite identificar a la persona.

2.9 MagicDraw UML

MagicDraw UML es una herramienta CASE compatible con el estándar UML para el diseño de software, y compatible con numerosos IDEs. Este programa se ha utilizado para la creación de los diagramas del diseño de nuestro sistema.

2.10 Pencil Project

Pencil Project es una interfaz gráfica de usuario (GUI) que permite, de forma gratuita y sencilla, crear prototipos de aplicaciones de escritorio y móviles. Se ha utilizado en esta ocasión para el diseño de bocetos de la interfaz gráfica de la aplicación Android, que podemos ver en el apartado 3.9 de este documento (correspondiente a la especificación de requisitos).

2.11 Trello

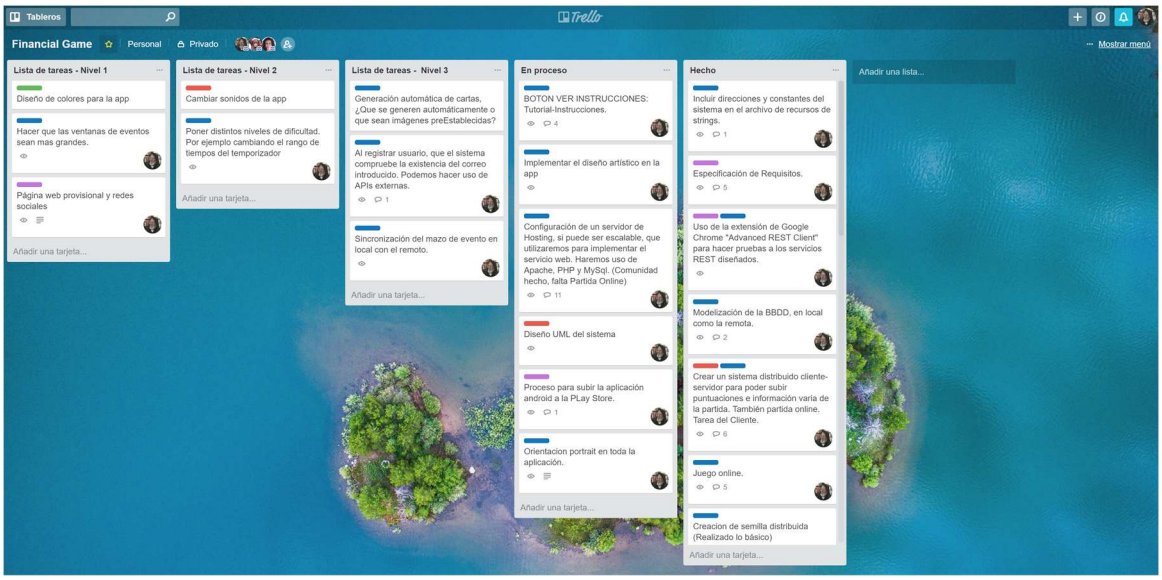


Figura: 9. Captura del portal web Trello

Trello es un software de administración de proyectos con interfaz web. También dispone de clientes para IOs y Android. La creación de una cuenta es totalmente gratuita, y podrás compartir el proyecto creado con otros usuarios. Con esta herramienta se han podido gestionar las distintas tareas que han ido surgiendo durante el transcurso del proyecto, así como tener un historial de las distintas actividades realizadas.

3 ESPECIFICACIÓN DE REQUISITOS

En este apartado, explicamos en detalle el análisis de los requisitos exigidos por el cliente, los objetivos a cumplir y la estructuración del sistema a desarrollar.

3.1 Introducción

Financial Game App es una aplicación móvil, desarrollada para Android, que complementa a un juego de mesa ya existente. Partiendo de éste y respetando su objetivo, introducirá nuevas posibilidades enriqueciendo la experiencia y actualizándola a los tiempos que corren. Con la aplicación los usuarios podrán; comprar productos financieros dentro del plazo marcado por un temporizador de tiempo aleatorio, almacenar sus datos, ver eventos dinámicos, crear gráficos de la partida y subir información a la comunidad online. Debemos indicar que únicamente se necesitará un dispositivo Android con la aplicación instalada, evitando la molestia de obligar a todos los jugadores, que comparten ubicación, a cumplir ese requisito. Por último, se implementará la funcionalidad estrella del programa, poder realizar partidas online con otros usuarios registrados en el sistema.

Para poder ofrecer todas las funciones online, será necesario implementar un servidor que ofrezca los servicios necesarios y satisfaga las peticiones de la aplicación Android, que realiza el rol de cliente.

Este proyecto será presentado como Trabajo Fin de Grado, en el cual se hará uso de los conocimientos aprendidos en la rama de telemática del Grado de Ingeniería de las Telecomunicaciones. Está organizado en distintos módulos, los cuales se irán explicando en este documento.

3.1.1 Propósito del documento

Este documento será de ayuda para entender las distintas partes de la aplicación software, los objetivos, quiénes interactúan con el software, cómo se comunican los distintos actores con la aplicación y los requisitos que debe satisfacer. Gran parte de los requisitos han surgido del proceso de análisis con el cliente, quien ha exigido que se respeten las ideas originales del producto que son la diversión y el aprendizaje. El papel de cliente lo ha realizado Antonio Escuder García, propietario del juego de mesa, creador de las reglas del juego y quien encargó la aplicación. Este documento servirá para establecer unas bases sobre las cuales desarrollaremos las siguientes fases.

3.1.2 Audiencia a la que va dirigido

Este documento va dirigido a todas las personas interesadas en entender el funcionamiento de la aplicación que se quiere desarrollar y comprender sus módulos, principalmente al cliente. También va dirigido a los diseñadores, desarrolladores y probadores del sistema.

3.1.3 Alcance

Lo primero de todo es dejar claro lo que hace y no hace el producto. La aplicación Android no será un sustituto del producto original (las reglas del juego en detalle están disponibles en el apartado *Situación de Partida*), ya que acompañará en todo momento al tablero y cartas. Será un complemento más, pero importante, para la partida. Los usuarios podrán con la aplicación; tener un temporizador de cada turno, ver el estado de las puntuaciones, ver eventos dinámicos que modificarán las variables del juego, introducir valores y ver un resumen de la partida finalizada. Tendrá características online como: un ranking de partidas y puntuaciones, sistema de registro y autenticación, sistema de logros, y ver información de perfil. Además, se introducirá el concepto de semilla distribuida, que consistirá en un escenario preestablecido que los usuarios conectados

podrán pedir al servidor para empezar la partida con unos valores concretos (dentro de un contexto económico), haciendo cada partida única. Todas estas tareas de gestión que podrá realizar el usuario las hemos englobado dentro de *Comunidad*, ya que es la forma por la cual el jugador podrá interactuar con la información ofrecida de otros jugadores y participar en una comunidad online que sostiene al juego de mesa. Por último, se implementará un modo de juego online.

Los usuarios podrán acceder a la aplicación desde la Play Store con un código que se encontrará dentro de la caja del juego de mesa, y deberán registrarse para poder acceder a las funciones online del programa. Por lo tanto, el producto final que recibirá el usuario estará compuesto por el juego de mesa y la aplicación Android que lo complementa, siendo nuestro objetivo el desarrollo esta última, el servidor que ofrece los servicios web, la modelización de la BBDD usada para almacenar la información y el despliegue.

3.1.4 actores

ACT-01	Usuario
Descripción:	Representa a los jugadores que harán uso de la aplicación móvil.
Comentario:	Su interacción se basará en visualizar e introducir valores.

ACT-02	BBDD remota
Descripción:	Representa la BBDD alojada en el servidor remoto.
Comentario:	Se alojará en servidores privados.

ACT-03	BBDD local
Descripción:	Representa a la BBDD almacenada localmente en el dispositivo móvil del usuario.
Comentario:	Necesaria para garantizar el correcto funcionamiento de la aplicación sin tener conexión a internet.

ACT-04	Archivo de preferencias
Descripción:	Forma alternativa de almacenar información de forma permanente en local. Lo utilizaremos para guardar parámetros de inicio de sesión y el identificador de la semilla distribuida.
Comentario:	Es un mecanismo liviano que permite almacenar y recuperar datos primitivos en forma de pares clave/valor. Un usuario, al identificarse, recibirá una clave de API necesaria para acceder a todos los servicios web, dicha clave se almacenará en

	preferencias.

3.2 Información del dominio del problema

Se parte de un juego de mesa, de temática financiera, que permite a las personas que se reúnen a su alrededor comprar y vender una serie de productos financieros imaginarios en un tiempo determinado, sufriendo los cambios de mercado producidos por eventos (representados por cartas) y por las mismas acciones de los jugadores (por ejemplo, vender una casa hará que el siguiente la tenga que vender más barata). Podrán pedir financiación, lo que supone más liquidez en el momento, pero intereses a pagar en el futuro, los cuales pueden ser catastróficos. Los Bonos del Estado y Plan de Pensiones ofrecerán un goteo continuo de capital, pero a costa de una inversión inicial importante. El recuento final de las puntuaciones será el resultado de la suma de todos los activos financieros de cada participante. El ganador será quien posea el mayor capital al final de la partida, o el último en no caer en banca rota (pierde toda la liquidez y no puede pedir más financiación).

Durante el transcurso de la partida, los jugadores estarán sujetos a un temporizador, obligando a planificar las acciones para comprar y vender lo más rápido posible, antes que el resto se adelanten. Esta característica dota de competitividad entre participantes. Las reglas del juego en detalle están disponibles en el siguiente apartado.

La aplicación Android propuesta complementará al juego de mesa descrito. Por lo tanto, el conjunto de conceptos referentes al dominio del problema será: usuarios, rondas, partida, temporizador, activos financieros, activos inmobiliarios, deuda, letras del estado, plan de pensiones, eventos, puntuación, ranking, ventana de resultados, servicio, ranking online, mazo de eventos, servicios, comunidad.

Hoy en día existen multitud de aplicaciones móviles de todo tipo, y no todas alcanzan el éxito. En nuestro caso es aún más complejo porque es una app que acompañará a un juego de mesa, y ambos elementos son necesarios para la partida. A esto debemos sumarle la implementación de características online que enriquecerán la experiencia con el producto en su conjunto. Esta es la principal característica de nuestro software, enriquecer la experiencia de juego gracias a las últimas innovaciones tecnológicas en la programación móvil, resultando una experiencia divertida y educativa.

3.3 Descripción de la situación de partida

Actualmente, Financial Game es un juego de mesa tradicional con el tablero, cartas, fichas e instrucciones. A continuación, se explica detalladamente las reglas del juego.

3.3.1 Reglas del juego

Los elementos físicos que componen el juego de mesa son: tablero, fichas del tablero (indican valores, 11 en total), cartas de evento (40 en total), cartas para cada tipo de producto financiero (20 para cada tipo, excepto para la financiación que sólo utiliza indicadores), billetes de papel para ayudar a los jugadores contabilizar su patrimonio (la moneda oficial del juego es el ₩) y un temporizador de arena. Será necesario que los participantes aporten una calculadora, papel y algo para escribir (no vienen incluidos).

El juego consta de cuatro rondas y de cuatro turnos dentro de cada ronda. La duración de los turnos lo indica el reloj de arena. Pueden participar un mínimo de dos personas y un máximo de seis. Al inicializar una partida se repartirán 100.000 ₩, se colocarán las respectivas cartas de productos financieros alrededor del tablero, y se barajará el mazo de eventos.

Dentro de cada turno, los jugadores, de forma simultánea, podrán comprar y vender los diferentes productos que se encuentran en el tablero: acciones de diferentes perfiles de rentabilidad y riesgo, viviendas, letras del tesoro y bonos del estado, así como invertir en plan de pensiones o financiarse. Para ello tan solo tienen que depositar en el banco (si compran o amortizan la deuda) o extraer (si venden o se financian) la cantidad a la que se puede adquirir cada producto en el mercado en ese momento.

En el momento que el reloj de arena acabe, se finalizará el turno y los jugadores pararán de realizar sus acciones para levantar la carta de evento superior del mazo. En dicho evento cambiarán las condiciones del mercado, surgiendo oscilaciones en el precio de los diferentes productos (unos se encarecerán y otros se abaratarán tal y como lo harían en condiciones reales). Una vez realizadas dichas modificaciones en el tablero central (con los marcadores correspondientes) se reanudará la partida con un nuevo turno. Además de eventos que cambian las condiciones del mercado, existen otros eventos de reparto de dividendos (a los que dispongan de determinado tipo de acciones en ese momento) o eventos inesperados de necesidades financieras, en los que se les exigirá a los jugadores disponer de efectivo para hacer frente a gastos inesperados (reparaciones en el hogar, necesidad de adquisición de vehículo, etc.)

Una vez se hayan realizado cuatro turnos, se finalizará la ronda y se procederá a realizar un recuento del patrimonio conseguido por cada jugador hasta ese momento. Para ello, se hará uso de una calculadora, y por orden, cada participante sumará el valor de los productos que posee (la financiación es una variable negativa). Es aconsejable apuntar las puntuaciones en un folio aparte, para evitar equivocaciones y olvidos.

De esta forma se sucederán las rondas y turnos hasta el final de la partida, momento en el que se compararán las puntuaciones totales de cada jugador. La persona con mayor puntuación (patrimonio) será la vencedora de la partida.

Durante el transcurso de la partida, una persona puede caer en banca rota si no tiene liquidez y además ya se ha financiado. Si alguien alcanza este estado, no puede continuar la partida y se le considera que ha perdido.

3.3.2 Conclusión

Aunque la temática en la que se basa le hace ser único, es cierto que su potencial se ve mermado por el ritmo lento que predomina en los tableros tradicionales, además de lo engorroso de tener una numerosa cantidad de cartas y la necesidad de que los jugadores apunten en papel sus acciones financieras. La petición del cliente es crear una aplicación móvil que solucione estos problemas, conectando las distintas personas que hayan comprado el producto. Somos conscientes lo complicado que es destacar en el mundo de las aplicaciones móviles y en el de los juegos de mesa si no existe detrás una comunidad de usuarios.

3.4 Necesidades de negocio

Los procedimientos que se utilizan para llevar el ritmo de la partida son rudimentarios y muy deficientes, peligrando una experiencia de juego positiva, provocando problemas como el olvido de las puntuaciones por parte de los usuarios, la invariación del tiempo del temporizador y la imposibilidad de conocer el progreso de un usuario durante el juego. Por ello estamos convencidos que la inclusión de una aplicación móvil aportará frescura al producto original, generando una experiencia divertida y enriquecedora.

Gracias a la app, los jugadores podrán tener a su disposición gran variedad de posibilidades, no sólo jugar a una partida offline. Podrían registrarse y participar en una comunidad online, compartir partidas, gestionar su usuario, realizar partidas online con otras personas y visualizar posibles logros conseguidos.

Referente al transcurso de la partida en sí, hay varios apartados a mejorar que contribuyen positivamente a la experiencia de juego:

- El número de elementos físicos que gestionar durante la partida. Si conseguimos implementar las cartas de evento en la aplicación, aliviamos el número de elementos que administrar, fomentando que los jugadores se centren en lo importante, jugar y disfrutar.
- Como está concebido el juego actualmente, los productos financieros siempre empiezan con los mismos valores, produciendo cierto desinterés en repetir. Con la aplicación, los participantes podrían elegir un escenario preestablecido, ya definido, que indicase la variación de valores del tablero dentro de un contexto económico. Con esto conseguiríamos mejorar la rejugabilidad.
- Tener un temporizador con el mismo valor siempre hace que a la larga el transcurso de los eventos sea monótono. Implementando este elemento en la aplicación, y dándole un tiempo aleatorio,

conseguiremos que la partida sea más ágil y divertida, ya que los participantes jugarán con cierta incertidumbre y reto.

- La aplicación podrá mostrar en todo momento un ranking actualizado de las puntuaciones de los jugadores, ofreciendo una información clave para hacer estrategias.
- Calcular manualmente la puntuación de cada jugador es un proceso tedioso, obligando a utilizar recursos propios (como la calculadora), además de romper el ritmo de la partida. Hacer que los jugadores puedan introducir sus datos y la app calcule automáticamente la puntuación sería un valor muy positivo que mejoraría la experiencia de juego drásticamente.
- Que llegue el fin de la partida y que únicamente se concluya con quién es el ganador, aporta poco valor al resto de los participantes. Con la aplicación se podría mostrar un historial de las distintas acciones de los jugadores y sus puntuaciones a lo largo de la partida. De esta manera, todos podrían aprender de sus errores para futuras sesiones.
- Almacenar las instrucciones en formato digital sería un gran avance para asegurar a los jugadores que siempre tendrán a su disposición las reglas de juego, y no preocuparse en que las instrucciones impresas puedan perderse o estropearse.

Además, la necesidad de diferenciarse en el mercado de juegos de mesa para poder crecer es fundamental, siendo el software creado una oportunidad estupenda. Actualmente muy pocos juegos de mesa aportan esta diferenciación, y aquellas aplicaciones Android basadas en juegos de mesa han perdido la esencia del producto original ya que pierde la relación física entre usuarios.

Por último, indicamos que las pruebas realizadas con y sin el software muestran claramente los beneficios y el valor añadido que la app aporta en el producto final, convirtiendo un juego de mesa tradicional en una experiencia más enriquecedora para los usuarios, queriendo jugar más de una partida seguida para intentar aprender de los errores y mejorar las puntuaciones.

3.5 Objetivos

OBJ-01	Partidas offline
Descripción:	La aplicación ofrecerá la posibilidad de realizar partidas offline entre usuarios registrados en el sistema.
Comentario:	La aplicación deberá almacenar localmente la información necesaria para asegurar que se pueda jugar sin conexión. No necesitará conexión continua con el servidor (se podrá ejecutar una partida offline de principio a fin sin conexión).

OBJ-02	Partidas online
Descripción:	La aplicación, gracias al uso de servicios web, ofrecerá la posibilidad de realizar partidas online entre usuarios registrados en el sistema.
Comentario:	Mantiene la esencia de las partidas offline, pero en este caso, se alternará entre los usuarios. También se deberá programar los respectivos servicios web ofrecidos por el servidor.

OBJ-03	Gestión del ritmo de la Partida
Descripción:	La aplicación llevará el ritmo de la partida, permitiendo iniciarla, indicar nombres, mostrar temporizador, mostrar eventos y permitir introducir puntuaciones.
Comentario:	Hay distintos tipos de eventos, y los usuarios podrán ver un ranking al introducir las puntuaciones.

OBJ-04	Representación del resumen de la partida
Descripción:	Al finalizar la partida se mostrará información en modo de resumen para que los jugadores puedan ver un historial de puntuaciones y eventos.
Comentario:	Se utilizarán varios tipos de gráficas.

OBJ-05	Gestión de acceso a los servicios de la comunidad
Descripción:	El usuario podrá visualizar en la aplicación toda la información referente a su perfil online y la comunidad.
Comentario:	La información podrá ser; partidas compartidas, obtener la "semilla distribuida", acceder al ranking online, identificación y recompensas. También se deberá programar los respectivos servicios web ofrecidos por el servidor.

OBJ-06	Consulta de instrucciones y manual de usuario
Descripción:	La aplicación Android ofrecerá la posibilidad de consultar las instrucciones del juego, y un manual de usuario que explique las principales funciones de la aplicación Android.
Comentario:	Serán documentos de texto embebidos en la aplicación.

3.6 Descripción de subsistemas a desarrollar

Se propone realizar un software con el que, a través de una interfaz, los usuarios pueden interactuar con las distintas funciones de la aplicación, que dan valor al producto. El programa ofrecerá la posibilidad de llevar el ritmo de la partida, participar en la comunidad, jugar partidas online y visualizar las instrucciones en formato digital.

El objetivo de la división de subsistemas que hemos realizado es facilitar el desarrollo, la integración y las pruebas, así como hacer más comprensible nuestro software. Hemos dividido la aplicación en cinco subsistemas bien diferenciados:

1. Gestión de partida offline.
2. Gestión de partida online.
3. Gestión de la comunidad.
4. Consulta de instrucciones y manual de usuario.
5. El servidor de la aplicación.

Todos los subsistemas son independientes entre sí, por lo que se podrán desarrollar fácilmente por separado, lo que evita conflicto en versiones y fallos de software. Sin embargo, algunas funciones de gestión de la comunidad (como subir al servidor una partida jugada) podrán ser accedidas desde gestión de partida offline u online. A continuación, se explica con más detalle cada subsistema:

- **Gestión de partida offline (OBJ-01, OBJ-03, OBJ-04):** Compuesto por todas las actividades y sus respectivas vistas, que permiten a los usuarios reunidos físicamente empezar una partida offline y terminarla. Los jugadores podrán; elegir opciones de la partida (la semilla distribuida entre ellas), gestionar el ritmo de la partida con un temporizador aleatorio, ver eventos, introducir valores de los distintos activos financieros de cada jugador, conocer el ganador y ver un resumen con gráficas de la partida. Sólo necesitará conectividad con el servidor para la función de compartir partida con la comunidad, que se ofrecerá al usuario tras finalizarla.
- **Gestión de partida online (OBJ-02, OBJ-03, OBJ-04):** Compuesto por todas las actividades y sus respectivas vistas, que permiten a los usuarios conectados empezar una partida online y terminarla. El jugador anfitrión (quien inicia una partida) podrá elegir opciones de la partida (la semilla distribuida entre ellas) y al final compartirla con el resto de la comunidad. El conjunto de usuarios conectados podrá gestionar el ritmo de la partida con un indicador de turno aleatorio, ver eventos, introducir valores de los distintos activos financieros de cada jugador, conocer el ganador y ver un resumen con gráficas de la partida. Implica el consumo de servicios web, que deberán ser desarrollados con anterioridad en el servidor. Se seguirá el paradigma REST.
- **Gestión de la comunidad (OBJ-05):** Se encargará de gestionar las funciones sociales del programa como; registrar un usuario, iniciar sesión, ver ranking de puntuaciones, modificar el perfil del usuario, ver el historial de partidas, sistema de logros y sistema de la semilla distribuida. Todos los servicios que parten de esas acciones se encargarán de almacenar, visualizar y modificar la información del servidor desde la aplicación (deberán ser desarrollados con anterioridad). Se seguirá el paradigma REST.
- **Consulta de instrucciones y manual de usuario (OBJ-06):** Proporciona la capacidad para mostrar la documentación correspondiente a las instrucciones de la partida, y el manual de usuario para explicar cómo se navega por la aplicación.
- **Servidor de la aplicación:** Se encargará de gestionar los distintos servicios ofrecidos por el servidor, y hará uso de un gestor de BBDD. Las distintas funciones se ofrecerán por medio de mensajes HTTP que usarán la estructura de datos JSON.

3.7 Diagramas de casos de uso

Como introducción a los siguientes apartados, mostraremos un diagrama de caso de uso generalista. En los próximos apartados iremos profundizando en cada uno de los subsistemas. Además, se incluyen las tablas de casos de uso para ofrecer más detalle.

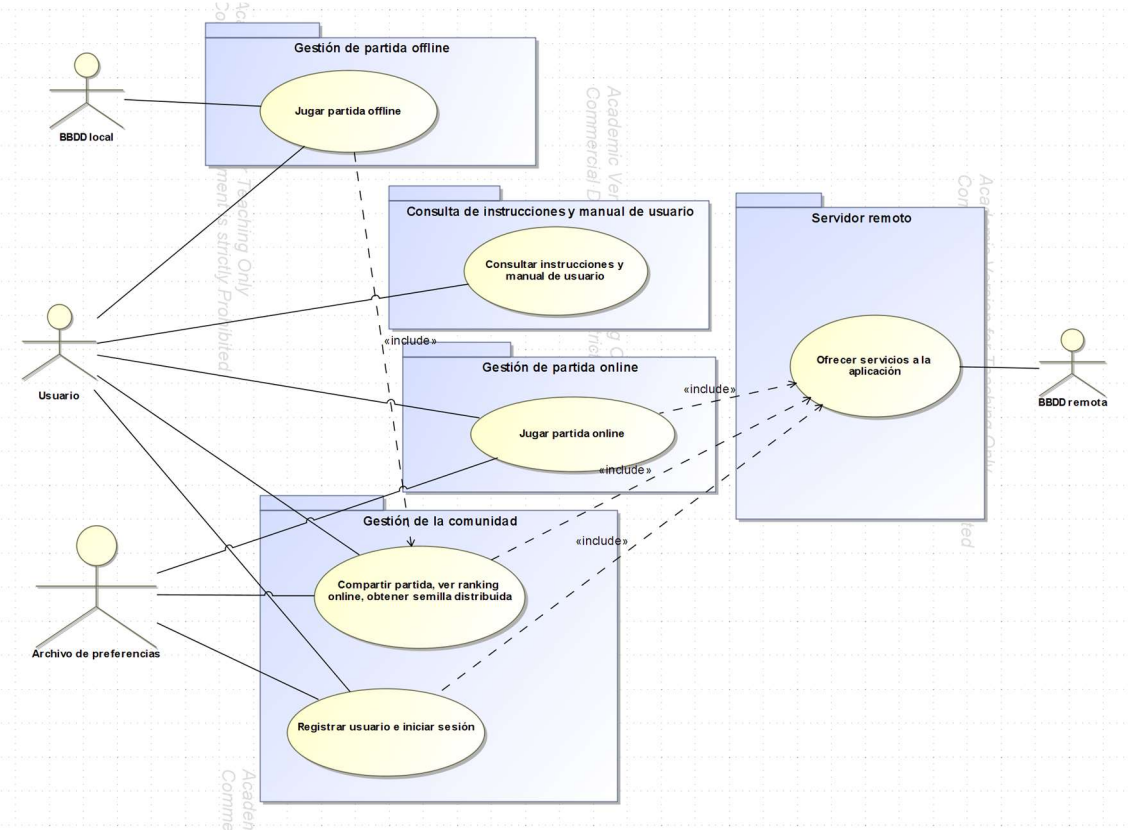


Figura: 10. Diagrama de casos de uso general del sistema

Debemos indicar que, cuando una persona inicia la aplicación, la primera pantalla mostrada es la pantalla de carga. Se indica en la siguiente tabla.

-	**Mostrar pantalla de carga**
Descripción:	Cuando una persona lanza la aplicación, la primera pantalla mostrada será la pantalla de carga, que tendrá una duración de tres segundo como máximo. En esta vista aparecerá un indicador de carga y el logo de la app.

3.7.1 Gestión de partida offline

A continuación, se muestra un diagrama de casos de uso referente al subsistema Gestión de partida offline:

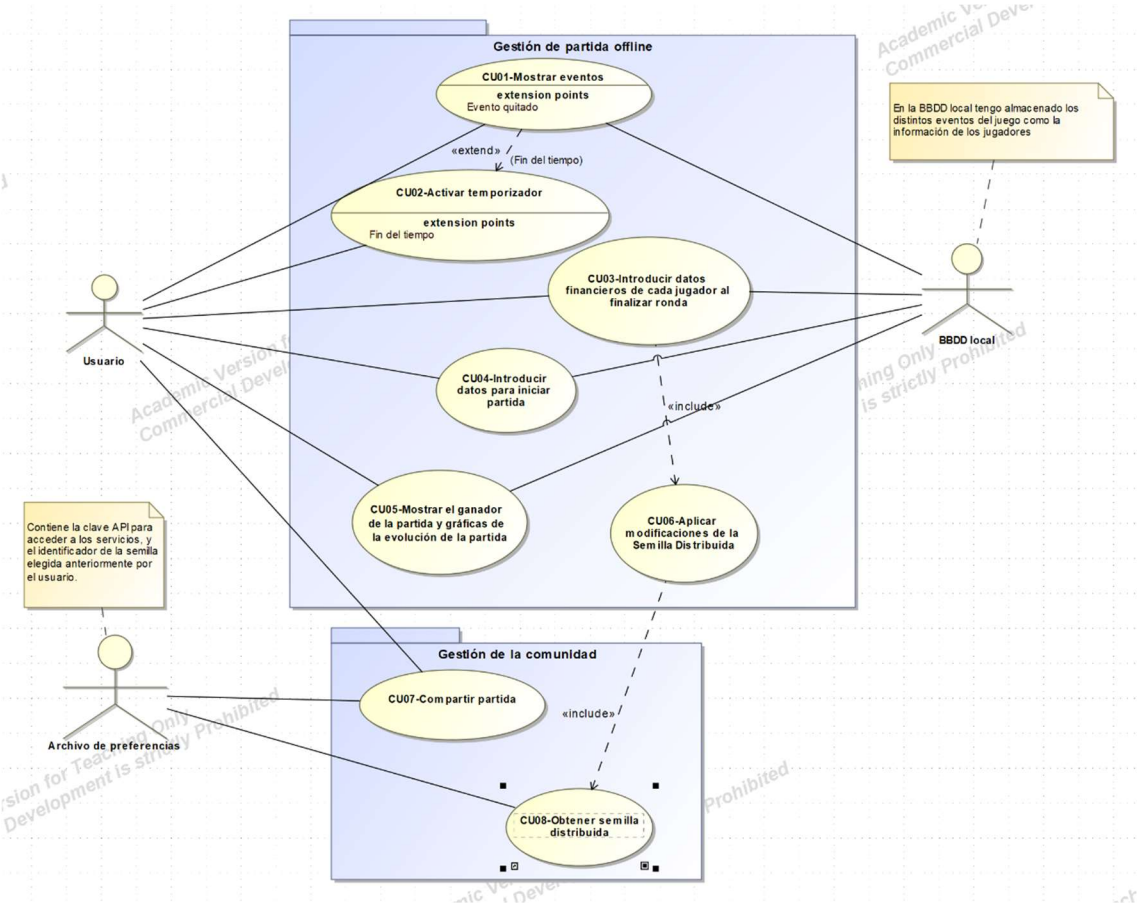


Figura: 11. Diagrama de casos de uso de partida offline

CU01	Mostrar eventos
Actores:	Usuario, BBDD local
Pre-condición:	Tiempo del temporizador acabado.
Descripción:	La aplicación mostrará una ventana emergente con la información del evento seleccionado. Éste lo obtiene de la BBDD local.
Post-condición:	El usuario verá el evento, y actuará en consecuencia de la información mostrada.

CU02	Activar temporizador
Actores:	Usuario
Pre-condición:	El usuario pulsa sobre el botón iniciar temporizador, o se acaba de cerrar una ventana de evento.
Descripción:	La aplicación ejecutará un temporizador de tiempo aleatorio. Se mostrará en pantalla un indicador del avance del tiempo.
Post-condición:	Al finalizar el tiempo, se ejecutará un aviso sonoro y mostrará automáticamente la ventana de evento.

CU03	Introducir datos financieros de cada jugador al finalizar ronda
Descripción:	Al finalizar la ronda, se mostrará la pantalla de introducir las puntuaciones correspondientes de cada jugador. Cuando se pulse en confirmar datos, se guardarán en la base de datos local. Cuando todos los participantes hayan escrito y confirmado sus puntuaciones, volveremos a la pantalla de la partida si no hemos terminado el juego. En caso de numero de rondas jugadas igual al número de rondas establecido, se mostrará el fin de la partida.

CU04	Introducir datos para iniciar partida
Descripción:	Antes de empezar una partida, la aplicación mostrará una pantalla donde los jugadores pueden configurar valores de la partida como número de rondas y los participantes. También ofrecerá la posibilidad de indicar si quiere utilizar la semilla distribuida en la partida.

CU05	Mostrar el ganador de la partida y gráficas de la evolución de la partida
Descripción:	El programa mostrará una ventana con información del jugador ganador. Los participantes podrán acceder a una pantalla con gráficas, pulsando el botón correspondiente, que representan el historial de puntuaciones de cada jugador.

CU06	Aplicar modificaciones de la Semilla Distribuida
Descripción:	En caso de que el jugador lo quiera, el sistema aplicará las modificaciones impuestas por la semilla distribuida seleccionada.

CU07	Compartir partida
Descripción:	El usuario, al darle al botón correspondiente, indicará a la aplicación que suba al servidor los datos de la partida finalizada.

CU08	Obtener semilla distribuida
Descripción:	El sistema manda una petición al servidor con el identificador de la semilla seleccionada, para obtener los datos de ésta. En caso de no enviar identificador, devolvería un listado con todas las semillas existentes.

3.7.2 Gestión de partida online

A continuación, se muestra un diagrama de casos de uso referente al subsistema Gestión de partida online:

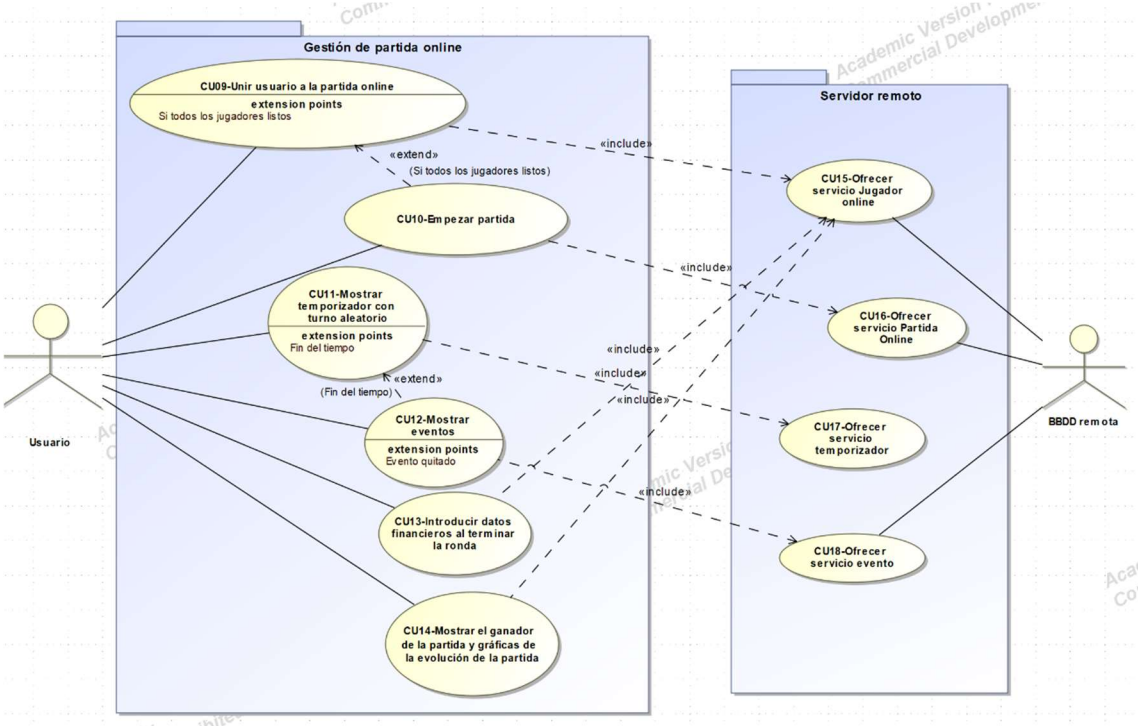


Figura: 12. Diagrama de casos de uso de partida online

CU09	Unir usuario a la partida online
Actores:	Usuario
Pre-condición:	El usuario selecciona una partida a la que unirse, o crea una partida online.
Descripción:	La aplicación manda una petición al servidor con el identificador de la partida para unir ese usuario con la partida online seleccionada.
Post-condición:	Usuario asignado y se mostrará la pantalla inicial de la partida online.

CU10	Empezar partida
Descripción:	Si todos los jugadores online están preparados, empezará automáticamente la partida, mostrándose la siguiente pantalla de este módulo.

CU11	Mostrar temporizador con turno aleatorio
Actores:	Usuario
Pre-condición:	Todos los usuarios preparados.
Descripción:	La aplicación mostrará en la pantalla de los dispositivos de todos los jugadores online el temporizador de tiempo aleatorio. Para ello realizará una petición al servidor para obtener el valor generado, el mismo para todos los participantes.
Post-condición:	Al finalizar el tiempo, se ejecutará un aviso sonoro y mostrará automáticamente la ventana de evento.

CU12	Mostrar eventos
Actores:	Usuario
Pre-condición:	Tiempo del temporizador acabado.
Descripción:	El sistema realizará una petición al servidor para obtener el evento aleatorio generado, el mismo para todos los participantes. La aplicación mostrará una ventana emergente con la información del evento obtenido.
Post-condición:	Todos los jugadores online visualizarán el mismo

	evento.

CU13	Introducir datos financieros al terminar la ronda
Descripción:	Al finalizar la ronda, se mostrará la pantalla de introducir las puntuaciones correspondientes de cada usuario. Cuando se pulse en confirmar datos, la aplicación mandará una petición con los datos introducidos para almacenarlos en la BBDD remota. Cuando todos los jugadores online estén preparados, volveremos a la pantalla de la partida si no hemos terminado el juego. En caso de numero de rondas jugadas igual al número de rondas establecido, se mostrará el fin de la partida

CU14	Mostrar el ganador de la partida y gráficas de la evolución de la partida
Descripción:	El programa mandará petición al servidor para conocer el ganador de la partida, y se lo mostrará a todos los jugadores online. Acto seguido lanzará la vista de gráficas que ofrece más información del transcurso de la partida.

CU15	Ofrecer servicio Jugador online
Descripción:	El servidor procesa las peticiones que recibe de los clientes respecto al recurso Jugador Online.

CU16	Ofrecer servicio Partida Online
Descripción:	El servidor procesa las peticiones que recibe de los clientes respecto al recurso Partida Online.

CU17	Ofrecer servicio temporizador
Descripción:	El servidor procesa las peticiones que recibe de los clientes respecto al recurso Temporizador.

CU18	Ofrecer servicio evento

Descripción:	El servidor procesa las peticiones que recibe de los clientes respecto al recurso Evento.

3.7.3 Gestión de la comunidad

A continuación, se muestra un diagrama de casos de uso referente al subsistema de Gestión de la comunidad:

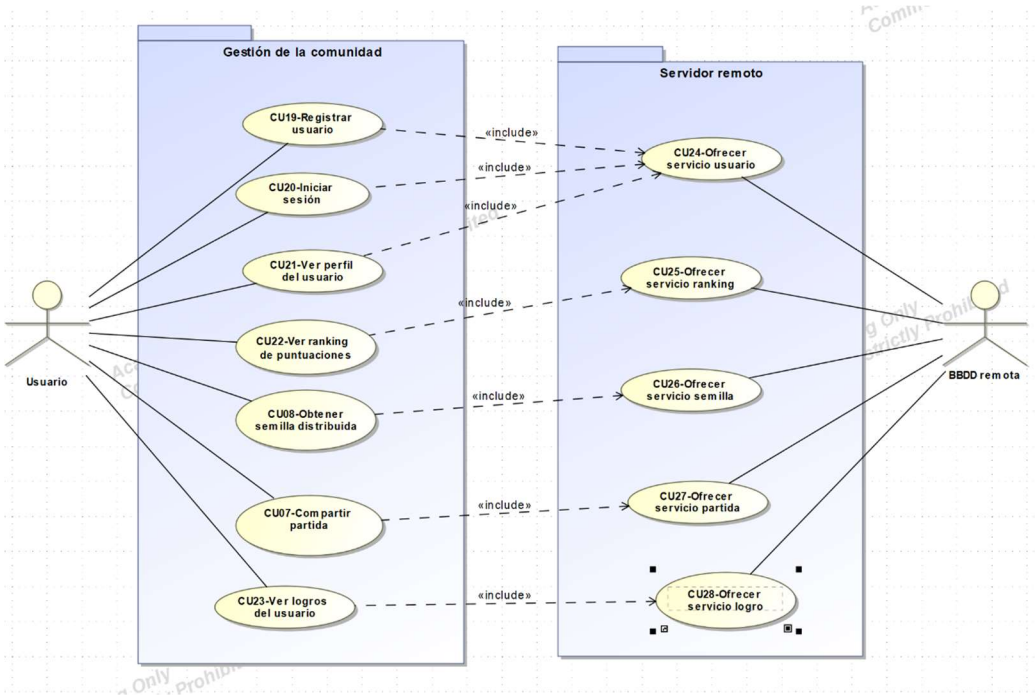


Figura: 13. Diagramas de casos de uso de comunidad

CU19	Registrar usuario
Descripción:	El usuario accede a la ventana emergente correspondiente a registrar usuario. La aplicación manda al servidor la petición de crear usuario con los datos introducidos.

CU20	Iniciar sesión
Descripción:	El usuario accede a la ventana emergente correspondiente a iniciar sesión. La aplicación manda al servidor la petición de identificar al usuario según los datos introducidos.

CU21	Ver perfil del usuario
Descripción:	El usuario accederá a la ventana de perfil, dentro de comunidad, y podrá ver su información

	personal. Para ello, la aplicación deberá mandar una petición al servidor preguntando por los datos de éste.

CU22	Ver ranking puntuaciones
Descripción:	El usuario accederá a la ventana de ranking, dentro de comunidad, y podrá ver una lista de las partidas con más puntuación final. Para ello, la aplicación deberá mandar una petición al servidor preguntando por las partidas subidas.

CU23	Ver logros del usuario
Descripción:	El usuario accederá a la ventana de logros, dentro de comunidad, y podrá ver una lista con sus logros obtenidos catalogados según el tipo. Para ello, la aplicación deberá mandar una petición al servidor preguntando por los logros del usuario.

CU24	Ofrecer servicio usuario
Descripción:	El servidor procesa las peticiones que recibe de los clientes respecto al recurso Usuario.

CU25	Ofrecer servicio ranking
Descripción:	El servidor procesa las peticiones que recibe de los clientes respecto al recurso Ranking.

CU26	Ofrecer servicio semilla
Descripción:	El servidor procesa las peticiones que recibe de los clientes respecto al recurso Semilla.

CU27	Ofrecer servicio partida
Descripción:	El servidor procesa las peticiones que recibe de los clientes respecto al recurso Partida.

CU28	Ofrecer servicio logro

Descripción:	El servidor procesa las peticiones que recibe de los clientes respecto al recurso Logro.

3.7.4 Consulta de instrucciones y manual de usuario

A continuación, se muestra un diagrama de casos de uso referente al subsistema de Consulta de instrucciones y manual de usuario:

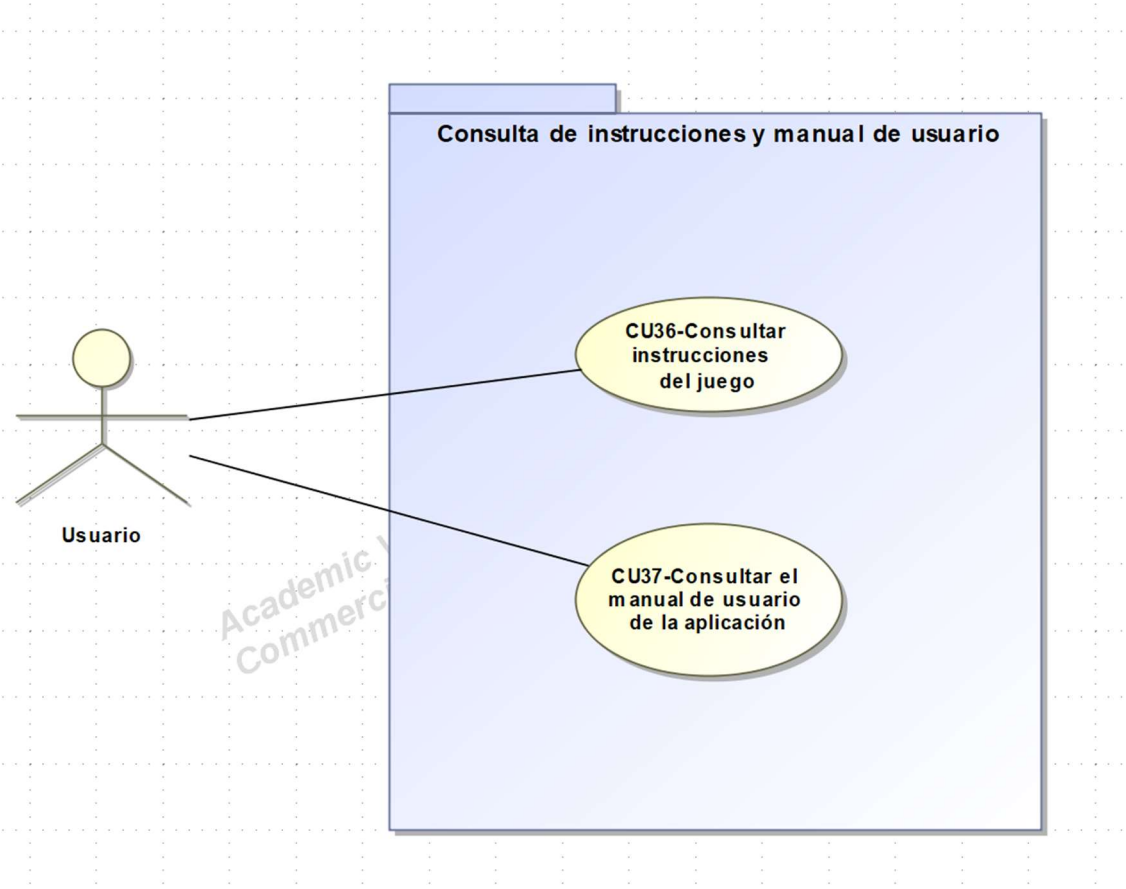


Figura: 14. Diagrama de casos de uso manuales de usuario

CU36	Consultar instrucciones del juego
Descripción:	El usuario podrá acceder a las instrucciones del juego a través de la sección correspondiente de la aplicación. Se mostrará un documento pdf.

CU37	Consultar el manual de usuario de la aplicación
Descripción:	El usuario podrá acceder al manual de usuario de la aplicación a través de la sección correspondiente. Se mostrará un documento pdf.

3.7.5 Servidor de la aplicación

El servidor desarrollado utiliza la arquitectura de software REST, por lo que ofrece representaciones de recursos. Éstos estarán identificados por URIs que el cliente deberá indicar en sus peticiones. La información enviada por el servidor estará representada en JSON, siendo tarea del cliente tratarla adecuadamente.

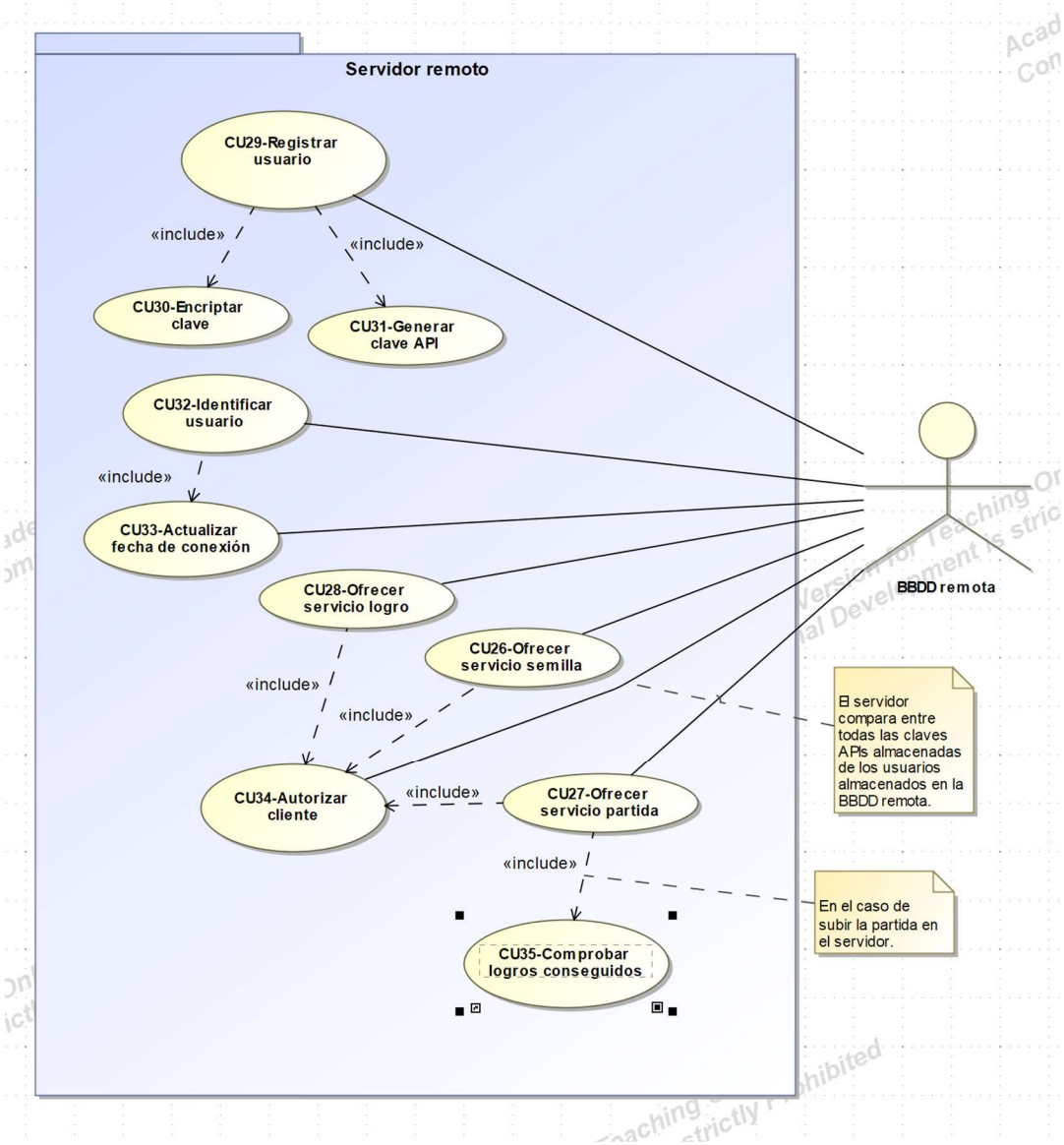


Figura: 15. Diagrama de casos de uso del servidor

CU29	Registrar usuario
Descripción:	El servidor registrará un nuevo usuario en la BBDD remota con la información recibida.

CU30	Encriptar clave
Descripción:	Al registrar un usuario, la contraseña introducida por él será encriptada en un código seguro.

CU31	Generar clave API
Descripción:	Cuando se registra un nuevo usuario en la BBDD remota, se le asociará una clave API necesaria para acceder a los servicios.

CU32	Identificar usuario
Descripción:	El servidor comprobará si existe en el sistema algún usuario con las credenciales introducidas (correo y contraseña). En caso positivo, mandará al cliente la clave API asociada.

CU33	Actualizar fecha de conexión
Descripción:	Cuando un usuario se identifica correctamente, el servidor actualizará la fecha de la última conexión de ese usuario.

CU34	Autorizar cliente
Descripción:	Antes de satisfacer la petición del cliente, éste deberá estar identificado. Para ello se cogerá el código API de la cabecera de la petición HTTP y se comparará con las almacenadas. En caso de que exista, se dará por autorizado.

CU35	Comprobar logros conseguidos
Descripción:	Cuando un usuario sube al servidor una partida, ésta se pasará por una serie de filtros para comprobar si cumple con algún requisito de los logros. En caso positivo, se relacionarán usuario y el logro conseguido.

3.8 Catálogo de requisitos a desarrollar

3.8.1 Requisitos de información

IRQ-01	Información del jugador de la partida
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-03, OBJ-04
Descripción:	Se guardará información en sobre los jugadores que participan en una partida.
Datos específicos:	• Nombre (Cadena de caracteres) • Puntuación (Entero) • Nº de Acciones (Entero) • Nº de Bienes Inmuebles (Entero) • Nº de Bonos del Estado (Entero) • Nº de Plan de Pensiones (Entero) • Efectivo (Entero) • Financiación (Entero)
Prioridad:	Alta
Comentario:	Tiene su respectiva tabla. En la modalidad offline se guardará en la BBDD local, pero en la modalidad online se guardará en la BBDD remota.

IRQ-02	Mazo de eventos
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-03
Descripción:	Se almacenará un conjunto de cartas (eventos) con una imagen y atributos asociados.
Datos específicos:	• Conjunto de evento (IRQ-3)
Prioridad:	Alta
Comentario:	Se almacenará en tres tablas distintas; mazo, carta, atributos. En la BBDD local habrá una copia del mazo situado en la BBDD remota.

IRQ-03	Evento
Versión:	Versión 2.0 <16-05-2018>

Objetivos asociados:	OBJ-03
Descripción:	Representa a un evento que se mostrará a los jugadores, compuesto por una serie de atributos y una imagen asociada al tipo de evento.
Datos específicos:	• Identificador de evento (Entero) • Nombre del evento (Cadena de caracteres) • Tipo de evento (Cadena de caracteres) • Atributos del evento (IRQ-05)
Prioridad:	Alta
Comentario:	Un evento hace referencia a elementos del tablero que los jugadores deberán modificar, o a productos financieros. Gracias al identificador de atributo, se podrá crear eventos fácilmente sólo asociándole atributos guardados en la BBDD. Da flexibilidad al futuro mantenimiento de los eventos.

IRQ-04	**Listado de eventos**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-03
Descripción:	Conjunto de eventos fijos definidos para la inicialización de la BBDD.
Datos específicos:	• 1, 'Variaciones mercado financiero', 'Escenario financiero', 1, 6, 10, 14, 17 • 2, 'Variaciones mercado financiero', 'Escenario financiero', 1, 6, 8, 14, 20 • 3, 'Variaciones mercado financiero', 'Escenario financiero', 2, 5, 9, 13, 19 • 4, 'Variaciones mercado financiero', 'Escenario financiero', 2, 5, 7, 13, 18 • 5, 'Variaciones mercado financiero', 'Escenario financiero', 2, 5, 7, 11, 15 • 6, 'Variaciones mercado financiero', 'Escenario financiero', 2, 5, 7, 11, 18 • 7, 'Variaciones mercado financiero', 'Escenario financiero', 1, 6, 8, 12, 15 • 8, 'Variaciones mercado financiero', 'Escenario financiero', 1, 6, 17, 0, 0 • 9, 'Variaciones mercado financiero', 'Escenario financiero', 2, 5, 8, 14, 15 • 10, 'Reparto dividendos', 'Reparto', 21, 22, 23, 0, 0 • 11, 'Reparto dividendos', 'Reparto', 24, 25, 26, 0, 0 • 12, 'Reparto dividendos', 'Reparto', 27, 28, 29, 0, 0

	• 13, 'Desempleo temporal', 'Necesidades financieras', 31, 0, 0, 0, 0 • 14, 'Gastos universitarios', 'Necesidades financieras', 31, 0, 0, 0, 0 • 15, 'Adquisicion de vehiculos', 'Necesidades financieras', 33, 0, 0, 0, 0 • 16, 'Reformas en el hogar', 'Necesidades financieras', 32, 0, 0, 0, 0 • 17, 'Viaje familiar', 'Necesidades financieras', 30, 0, 0, 0, 0
Prioridad:	Alta
Comentario:	-

IRQ-05	**Atributo**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-03
Descripción:	Texto que hace referencia a un elemento en concreto del tablero, o a un producto financiero.
Datos específicos:	• Identificador de atributo. (Entero) • Descripción. (Cadena de caracteres)
Prioridad:	Alta
Comentario:	-

IRQ-06	**Listado de atributos**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-03
Descripción:	Conjunto de atributos fijos definidos para la inicialización de la BBDD.
Datos específicos:	• 1, Tipo interes variable ↥ • 2, Tipo interes variable ↧ • 3, Tipo interes variable ↥↥ • 4, Tipo interes variable ↧↧ • 5, Precio mercado inmobiliario ↥ • 6, Precio mercado inmobiliario ↧ • 7, Cotizacion acciones europeas ↥ • 8, Cotizacion acciones europeas ↧ • 9, Cotizacion acciones europeas ↥↥ • 10, Cotizacion acciones europeas ↧↧

	• 11, Cotizacion acciones americanas ↥ • 12, Cotizacion acciones americanas ↧ • 13, Cotizacion acciones americanas ↥↥ • 14, Cotizacion acciones americanas ↧↧ • 15, Cotizacion acciones emergentes ↥ • 16, Cotizacion acciones emergentes ↧ • 17, Cotizacion acciones emergentes ↥↥ • 18, Cotizacion acciones emergentes ↧↧ • 19, Cotizacion acciones emergentes ↥↥↥ • 20, Cotizacion acciones emergentes ↧↧↧ • 21, Acciones europeas 3000 • 22, Acciones americanas 5000 • 23, Acciones emergentes 7000 • 24, Acciones europeas 7000 • 25, Acciones americanas 3000 • 26, Acciones emergentes 5000 • 27, Acciones europeas 5000 • 28, Acciones americanas 7000 • 29, Acciones emergentes 3000 • 30, Necesidad al terminar la ronda 5000 • 31, Necesidad al terminar la ronda 10000 • 32, Necesidad al terminar la ronda 15000 • 33, Necesidad al terminar la ronda 20000
Prioridad:	Alta
Comentario:	-

IRQ-07	**Número de rondas**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-03, OBJ-04
Descripción:	La aplicación almacenará el número de rondas introducido por los usuarios en la inicialización de la partida.
Datos específicos:	• Número de rondas. (Entero)
Prioridad:	Alta
Comentario:	Se ofrecerá a los jugadores la posibilidad de introducir la duración de la partida, por lo que deberán indicar un número de rondas comprendido entre 3 y 6, ambos incluidos (antes este valor era siempre 4). Cuando se haya igualado el número de rondas, se habrá acabado el juego.

IRQ-08	**Número de turnos por cada ronda**

Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-03
Descripción:	El sistema tendrá almacenada una variable con el número de turnos que tendrá cada ronda. Tendrá un valor fijo de 4, sin ser modificable por los jugadores.
Datos específicos:	• Número de turnos por ronda (Entero)
Prioridad:	Alta
Comentario:	Cuando finalice el turno, se mostrará un evento, y cuando se haya igualado el número de turnos, se habrá acabado una ronda.

IRQ-09	**Intervalo de tiempo para el temporizador**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-03
Descripción:	EL sistema tendrá almacenada el intervalo de tiempo el cual se encuentra el temporizador.
Datos específicos:	• Tiempo mínimo (Entero) • Tiempo máximo (Entero)
Prioridad:	Alta
Comentario:	Se cogerá un valor aleatorio comprendido en ese intervalo.

IRQ-10	**Información del usuario**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-05
Descripción:	EL servidor de la app almacenará información en su BBDD todos los usuarios registrados.
Datos específicos:	• Identificador de usuario (Entero) • Nombre (Cadena de caracteres) • Contraseña (Cadena de caracteres) • Código de acceso a los servicios (Cadena de caracteres) • Fecha de la última conexión (Fecha) • Número de partidas jugadas (Entero) • Correo (Cadena de caracteres)

	• Logros (IRQ-13) • Semilla (IRQ-12) • Partidas del usuario (IRQ-11)
Prioridad:	Alta
Comentario:	Se ofrecerán servicios para que el usuario pueda acceder a su perfil. Para ello se le identificará con un código de acceso.

IRQ-11	**Información de la partida**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-05
Descripción:	El servidor almacenará información de las últimas partidas subidas.
Datos específicos:	• Identificador de la partida (Cadena de caracteres) • Identificador de usuario asociado (Entero) • Nº de Jugadores (Entero) • Fecha (Fecha) • Puntuación del ganador (Entero) • Nombre del ganador (Cadena de caracteres) • Nº de rondas (Entero) • Identificador de la semilla utilizada (Entero)
Prioridad:	Alta
Comentario:	Usuario asociado es quien se identificó en el sistema y subió la partida. Al jugar una partida, los jugadores podrán decidir si utilizar una semilla, en ese caso se indicará su identificador en la información de la partida.

IRQ-12	**Semilla distribuida**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-01, OBJ-02, OBJ-03
Descripción:	El servidor almacenará información sobre la semilla distribuida.
Datos específicos:	• Identificador de la semilla (Entero) • Título (Cadena de caracteres) • Descripción (Cadena de caracteres) • Variable afectada (Cadena de caracteres)

	• Valor (Entero)
Prioridad:	Media
Comentario:	La app cliente hará uso del servicio correspondiente para pedirla. La semilla afectará al valor de los activos financieros. Esta nueva característica se representará a los usuarios como "Escenario preestablecido".

IRQ-13	**Logro**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-01, OBJ-02, OBJ-03
Descripción:	El servidor almacenará información sobre los logros disponibles.
Datos específicos:	• Identificador del logro (Entero) • Nombre (Cadena de caracteres) • Descripción (Cadena de caracteres) • Tipo (Cadena de caracteres)
Prioridad:	Media
Comentario:	Cuando un usuario sube una partida al servidor, se le pasará por una serie de filtros para comprobar si ha cumplido algún logro. En ese caso, a ese usuario se le asocia el identificador del logro. Sólo hay tres tipos de logros: bronce, plata y oro. Esta función es nueva respecto a las reglas del juego original.

IRQ-14	**Manuales de la aplicación**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-06
Descripción:	La aplicación almacenará la documentación referente a los distintos manuales para poder jugar y entender cómo se usa la app.
Datos específicos:	• Instrucciones de la partida • Manual de usuario de la aplicación
Prioridad:	Alta
Comentario:	Serán archivos de texto.

3.8.2 Requisitos de reglas de negocio

RN-01	Turnos por ronda
Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	IRQ-08
Descripción:	Los usuarios no podrán modificar el número de eventos por ronda. El valor estará preestablecido por el sistema.
Prioridad:	Alta
Comentario:	Deberá tener un valor que no obstaculice la fluidez de la partida, comprendido entre 2 y 4 eventos.

RN-02	Intervalo del temporizador
Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	IRQ-09
Descripción:	Los usuarios no podrán modificar el intervalo de tiempo del temporizador. El intervalo estará preestablecido por el sistema. Antes de iniciar el temporizador, se cogerá un valor aleatorio comprendido en ese intervalo.
Prioridad:	Alta
Comentario:	Los valores mínimos y máximos no deben de diferenciarse en más de 30 segundos.

RN-03	Partidas simultáneas
Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	IRQ-01
Descripción:	Un mismo usuario sólo podrá participar en una partida a la vez (ya sean online u offline)
Prioridad:	Alta
Comentario:	Si el usuario sale de la ventana de una partida online en transcurso, será expulsado de ésta.

RN-04	Configurar los valores iniciales de la partida

Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	IRQ-11, IRQ-12
Descripción:	Los usuarios podrán elegir los valores de número de rondas, las personas que jugarán (sólo para partida offline), y si quieren utilizar alguna semilla antes de empezar la partida.
Prioridad:	Alta
Comentario:	-

RN-05	**Ver el temporizador en el transcurso de la partida**
Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	IRQ-09
Descripción:	Los jugadores podrán visualizar el temporizador del juego que aparece entre los distintos eventos. Aparecerá una ventana emergente con un indicador del progreso del temporizador.
Prioridad:	Alta
Comentario:	El tiempo del temporizador no es conocido ya que se coge un valor aleatorio de un intervalo.

RN-06	**Aviso al introducir las puntuaciones tras finalizar la ronda**
Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	IRQ-01
Descripción:	Al confirmar los datos introducidos por un usuario (en la vista correspondiente de introducir las puntuaciones de cada jugador), el sistema deberá mostrar un mensaje de aviso preguntando si está seguro de que los datos son correctos. Se le ofrecerá la posibilidad de modificarlos.
Prioridad:	Alta
Comentario:	Con este requisito nos aseguramos de que no haya datos introducidos accidentalmente.

RN-07	Ver evento
Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	IRQ-03
Descripción:	Se mostrará un evento al finalizar el temporizador. Se mostrará en una ventana emergente de la vista de la aplicación. En ella se verá: una imagen ilustrativa del tipo de evento, el tipo de evento, y atributos.
Prioridad:	Alta
Comentario:	El evento corresponde a una carta del mazo.

RN-08	Introducir puntuación de los jugadores
Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	OBJ-03
Descripción:	Al terminar una ronda, los jugadores deberán introducir en el sistema sus puntuaciones.
Prioridad:	Alta
Comentario:	Si no se indica un valor en un campo, por defecto será cero.

RN-09	Ver un resumen final de la partida
Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	IRQ-11
Descripción:	Los jugadores podrán ver al final de la partida un resumen de ésta. Se utilizarán gráficas para mostrar el historial de puntuaciones totales de cada jugador, y para representar los distintos porcentajes de la puntuación final de cada jugador según el tipo de producto financiero comprado.
Prioridad:	Alta
Comentario:	Después de esta vista, el usuario podrá salir definitivamente de la partida. Se reiniciarán todos los valores.

RN-10	Servicio de registro y autenticación
Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	IRQ-10
Descripción:	El usuario podrá registrarse en el servidor del sistema. También podrá iniciar sesión en la aplicación para acceder a los servicios online.
Prioridad:	Media
Comentario:	Al registrarse podrá introducir un nombre de usuario, contraseña y correo electrónico.

RN-11	Comprobar correo
Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	IRQ-10
Descripción:	La aplicación al introducir un correo para registrar al usuario comprobará que tenga un formato válido.
Prioridad:	Baja
Comentario:	-

RN-12	Servicio de compartir partida jugada
Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	IRQ-11
Descripción:	Un usuario podrá subir al servidor una partida recientemente acabada (ya sea online u offline). Estará asociada a este usuario.
Prioridad:	Alta
Comentario:	Se subirá una instancia del modelo correspondiente a la partida.

RN-13	Servicio de ranking
Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	IRQ-11

Descripción:	Un usuario acreditado podrá ver un ranking de las puntuaciones obtenidas por los usuarios en las partidas subidas.
Prioridad:	Alta
Comentario:	Ordenado de mayor a menor.

RN-14	**Servicio de ver perfil del usuario**
Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	IRQ-10
Descripción:	Un usuario acreditado podrá ver su información de perfil.
Prioridad:	Media
Comentario:	Esta información podrá ser el correo, nombre de usuario, ultimas partidas subidas, contraseña y máxima puntuación obtenida.

RN-15	**Cambiar contraseña**
Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	IRQ-10
Descripción:	El usuario podrá cambiar su contraseña en la sección ver perfil de la aplicación.
Prioridad:	Baja
Comentario:	-

RN-16	**Servicio de logros**
Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	IRQ-13
Descripción:	El sistema, según el número de partidas jugadas, puntuación máxima conseguida y posición del ranking global, le ofrecerá una serie de recompensas en forma de medallas.
Prioridad:	Baja

Comentario:	Las medallas serán de bronce, plata y oro. No tendrán impacto en el transcurso de la partida. Se verá reflejado en el perfil del usuario.

RN-17	**Conseguir un logro**
Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	IRQ-13, IRQ-11
Descripción:	Un usuario no podrá tener copias de un mismo logro, si lo ha conseguido con anterioridad no podrá volver a obtenerlo.
Prioridad:	Baja
Comentario:	El servidor, antes de asignarle al usuario un logro, comprueba que no lo tenga ya asignado.

RN-18	**Servicio de obtener semilla distribuida**
Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	IRQ-12
Descripción:	El sistema ofrecerá la posibilidad a los jugadores de aplicar la semilla distribuida en la partida. Este elemento modificará variables de la partida.
Prioridad:	Media
Comentario:	Además de las modificaciones, también irá acompañado de una breve descripción para introducir a los jugadores en un contexto referente al mercado financiero. Alguna semilla sólo ofrece una descripción de nuevas reglas de juego, o variar el valor inicial de alguna variable del tablero.

RN-19	**Aplicar semilla distribuida**
Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	IRQ-11, IRQ-12
Descripción:	El usuario sólo podrá seleccionar una semilla a la vez. Por lo tanto, la aplicación sólo aplicará una semilla como máximo en la partida. No podrá cambiar de semilla en el transcurso de la partida.

Prioridad:	Baja
Comentario:	Si selecciona una nueva semilla, la nueva sustituirá a la antigua.

RN-20	**Consulta de manuales**
Versión:	Versión 2.0 <16-05-2018>
Requisitos asociados:	IRQ-14
Descripción:	Los usuarios podrán visualizar las instrucciones del juego y también el manual de usuario para aprender a usar la aplicación.
Prioridad:	Alta
Comentario:	La información se ofrecerá en forma de documento de texto.

3.8.3 Requisitos funcionales de conducta

RNFC-01	**Gestión de errores del acceso a los servicios**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-02, OBJ-05
Descripción:	Cada vez que se produzca un error al acceder a los servicios desde el cliente, se mostrará un mensaje en la vista de la aplicación con una breve descripción del error.
Prioridad:	Alta
Comentario:	El usuario no maneja los errores, esta acción corresponde al sistema.

RNFC-02	**Barajar las cartas de evento del mazo**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-03
Descripción:	Cuando la partida es iniciada, el sistema reordenará aleatoriamente el mazo de eventos almacenado en local. En caso de la partida online, esta acción la realiza el servidor.

Prioridad:	Alta
Comentario:	En ambos modos de juego, se instancia una lista que equivale al mazo y re reordena los índices de las cartas.

RNFC-03	**Mostrar la evolución de la partida durante el transcurso de ésta.**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-03
Descripción:	En la vista correspondiente al temporizador, se ofrecerá un ranking de los distintos jugadores de la partida y su puntuación.
Prioridad:	Alta
Comentario:	Ordenados de mayor a menor. Facilita a los jugadores conocer la situación de la partida en ese momento.

RNFC-04	**Actualización del mazo local**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-05
Descripción:	Se le ofrecerá al usuario identificado en el sistema, la posibilidad de actualizar el mazo de cartas almacenado en local.
Prioridad:	Baja
Comentario:	La aplicación cliente pregunta por la versión del mazo de eventos en el servidor. SI es mayor que la local, lo sustituye.

3.8.4 Requisitos no funcionales de fiabilidad

RNFF-01	**Inicio de sesión del usuario**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-05
Descripción:	Antes de iniciar sesión por parte del usuario, se le pedirá que se identifique con el correo y

	contraseña. En caso positivo, se procede a iniciar sesión, y se le enviará su código API. Para acceder a los distintos servicios, siempre se preguntará por el código API.
Prioridad:	Alta
Comentario:	En caso de que el usuario no haya iniciado sesión, no podrá acceder a las secciones online de la app, y se mostrará un mensaje ofreciendo que inicie sesión.

RNFF-02	**Procesos en segundo plano de la aplicación**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-01, OBJ-02, OBJ-05
Descripción:	Se asegurará que la aplicación no quedará congelada durante cualquier proceso.
Prioridad:	Alta
Comentario:	Por ejemplo, el temporizador será un proceso en segundo plano.

RNFF-03	**Mensajes de error de la aplicación**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-02, OBJ-05
Descripción:	El servidor responderá al cliente cuando se ha producido un error al acceder a un servicio.
Prioridad:	Alta
Comentario:	El cliente deberá tener distintas categorías de errores, según ha sido un error de acceso a la base de datos, o una URI mal formada, etc.

3.8.5 Requisitos no funcionales de usabilidad

RNFU-01	**Acceso a las funciones principales de la App**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-01, OBJ-2, OBJ-5, OBJ-06
Descripción:	Al iniciar la aplicación, el usuario podrá acceder a

	las principales funciones del sistema pulsando una opción, sin necesidad de entrar por medio de otras vistas. Estas funciones son: iniciar partida offline, iniciar partida online, entrar en la comunidad del sistema, iniciar sesión y ver los manuales (instrucciones y manual de usuario).
Prioridad:	Alto
Comentario:	Esta vista hará la función de menú principal.

RNFU-02	**Ingreso de datos**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-03
Descripción:	Los usuarios podrán introducir los valores de sus recursos financieros a base de indicar el número de éstos, sin introducir valores totales y sin necesidad de calcularlo anteriormente.
Prioridad:	Alta
Comentario:	Antes, podrán indicar el valor de recursos financieros comunes a todos.

RNFU-03	**El temporizador**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-03
Descripción:	Al empezar una ronda, los jugadores podrán iniciar el temporizador pulsando un botón. En caso de que no haya terminado la ronda, al quitar el evento se activará automáticamente.
Prioridad:	Alta
Comentario:	-

RNFU-04	**Las gráficas**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-04
Descripción:	Los usuarios sólo podrán acceder a las gráficas al finalizar la partida. En la vista donde se muestran,

	sólo se mostrará un simultáneamente, pudiendo acceder a las demás por menús o botones.
Prioridad:	Alta
Comentario:	-

RNFU-05	**Vista de la comunidad**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-05
Descripción:	El usuario identificado podrá, haciendo uso de un menú, acceder a las distintas funciones de la comunidad.
Prioridad:	Media
Comentario:	Las distintas funciones se mostrarán en fragmentos situados dentro de la vista de comunidad.

RNFU-06	**Iniciar partida online**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-02
Descripción:	El usuario que entra en este módulo de la aplicación tendrá dos opciones; apuntarse a una partida creada o crear una partida.
Prioridad:	Alta
Comentario:	Las partidas creadas estarán organizadas en una tabla con scroll vertical, y cada una indicará quién la ha creado, número de rondas y el número de jugadores unidos.

RNFU-07	**Retorno al menú principal**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-01, OBJ-02, OBJ-5, OBJ-06
Descripción:	Los jugadores podrán volver directamente al menú principal al finalizar una partida o desde el menú de la comunidad.

Prioridad:	Alta
Comentario:	-

3.8.6 Requisitos no funcionales de eficiencia

RNFE-01	**Tiempo de respuesta**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-05
Descripción:	La información mostrada en la sección Comunidad de la aplicación, deberá mostrarse en menos de 5 segundos.
Prioridad:	Alta.
Comentario:	Hemos tomado este valor de referencia ya que es el tiempo máximo que permite Android a una operación en el hilo principal, para que ésta no de sensación de bloqueo al usuario.

RNFE-02	**Disponibilidad del servidor**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-02, OBJ-05
Descripción:	El servidor deberá estar disponible todos los días de la semana, permitiendo un máximo de dos horas diarias de mantenimiento.
Prioridad:	Media
Comentario:	Será un requisito importante a la hora de elegir un Hosting para nuestro servidor.

RNFE-03	**Relación entre tiempo de juego y tiempo introduciendo los datos**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-03
Descripción:	El tiempo máximo necesario para que los distintos jugadores introduzcan sus puntuaciones no podrá ser mayor a la mitad de la duración de una ronda.

Prioridad:	Alta
Comentario:	Se deberá cronometrar para comprobar que se cumpla. Debemos tener en cuenta que variará según el número de participantes.

3.8.7 Requisitos no funcionales de seguridad

RNFS-01	**Información de perfil**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-05
Descripción:	Se deberá asegurar que un usuario sólo podrá acceder a su información personal, sin poder ver la del resto de usuarios registrados en el sistema.
Prioridad:	Alta
Comentario:	Se utilizará la clave API única de cada usuario.

RNFE-02	**Acceso a los servicios**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-02, OBJ-05
Descripción:	Las peticiones de los clientes al servidor deberán ir cifradas, ya que se envía información confidencial. Hacer uso del protocolo https.
Prioridad:	Media
Comentario:	Para ello, debemos elegir un Hosting que cumpla con este requisito.

RNFE-03	**Permisos de acceso a los servicios**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-02, OBJ-05
Descripción:	El cliente deberá indicar en la cabecera del mensaje http la clave API asociada al usuario identificado. En ningún caso se tramitará una petición que no contiene clave API o contiene una que no existe.

Prioridad:	Alta
Comentario:	Para registrar un usuario, e iniciar sesión, no será necesario enviar la clave API.

3.8.8 Requisitos no funcionales de mantenibilidad

RNFM-01	Mantenibilidad
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-02, OBJ-05
Descripción:	Se podrá gestionar la BBDD y los distintos servicios directamente a través del servidor.
Prioridad:	Alta
Comentario:	En este caso, se utilizan las herramientas ofrecidas por el proveedor hosting.

3.8.9 Restricciones técnicas

RT-01	Sistema distribuido
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-02, OBJ-05
Descripción:	El sistema deberá utilizar la arquitectura cliente-servidor para gestionar los distintos servicios online. El rol de cliente será realizado por la aplicación móvil, y el servidor estará alojado en un hosting de terceros.
Prioridad:	Alta
Comentario:	En la versión Beta del sistema, el servidor estará alojado en 000webhost con una promoción gratuita.

RT-02	Uso de arquitecturas de software
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	-
Descripción:	El software deberá estar desarrollado siguiendo la

	arquitectura Modelo Vista Controlador. Además, las clases destinadas a acceder a los servicios deberán implementar Singleton.
Prioridad:	Alta
Comentario:	Las vistas serán los layout de la app, los controladores serán las activities y servicios, y el modelo estará compuesto por las clases de las cuales se aplican los servicios.

RT-03	**Tipo de servidor**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-02, OBJ-05
Descripción:	EL servidor deberá contener los siguientes componentes: Apache + MySQL + PHP
Prioridad:	Alta
Comentario:	El proveedor de sitios web 000webhost cumple este requisito ya que ofrece esas herramientas en sus servidores gratuitos.

RT-04	**Tipo de aplicación**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	-
Descripción:	La aplicación deberá ser Android nativa.
Prioridad:	Alta
Comentario:	Será desarrollada con Android Studio.

RT-05	**Conexión con el servidor**
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	OBJ-02, OBJ-05
Descripción:	Para acceder a los servicios se hará uso de REST.
Prioridad:	Alta

Comentario:	Esto es debido a que es orientado a recursos y tiene mayor compatibilidad con sistemas móviles.

3.8.10 Requisitos de integración

RI-01	Integración con librerías externas
Versión:	Versión 2.0 <16-05-2018>
Objetivos asociados:	-
Descripción:	El sistema por desarrollar deberá ser capaz de utilizar librerías externas, modificando el archivo de configuración correspondiente del proyecto Android.
Prioridad:	Alta
Comentario:	Por ejemplo, para la representación de gráficas, y el acceso a los servicios web para hacer peticiones http y gestionar la respuesta.

3.9 Interfaz gráfica

La interfaz gráfica de nuestro software está pensada para ser una herramienta útil e intuitiva para los jugadores. Gracias a la interfaz, los usuarios podrán acceder a los distintos módulos de la aplicación de forma rápida y sencilla. Para ello, el diseño gráfico elegido deberá ser minimalista, acorde con la temática tratada (las financias).

La división e interfaces se hará de la siguiente manera:

- Vista de inicio y menú principal.
- Modos de juego online y offline.
- Gestión de la comunidad.
- Visualización de manuales.

3.9.1 Vista de inicio y menú principal

Cuando el usuario inicie la aplicación, lo primero que verá será una vista de carga (SplashScreen) con el logo y nombre del programa, en esta vista el usuario no podrá interactuar y deberá esperar. A continuación, aparecerá el menú principal, compuesto por los botones que mandarán al usuario a los distintos módulos de la app.

También se ofrecerá un botón de acceso directo para iniciar sesión desde el menú. A continuación, se muestran unos bocetos de estas interfaces:

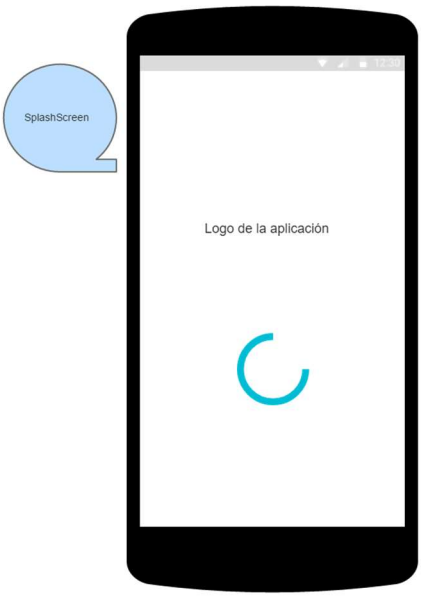


Figura: 16. Boceto vista de carga

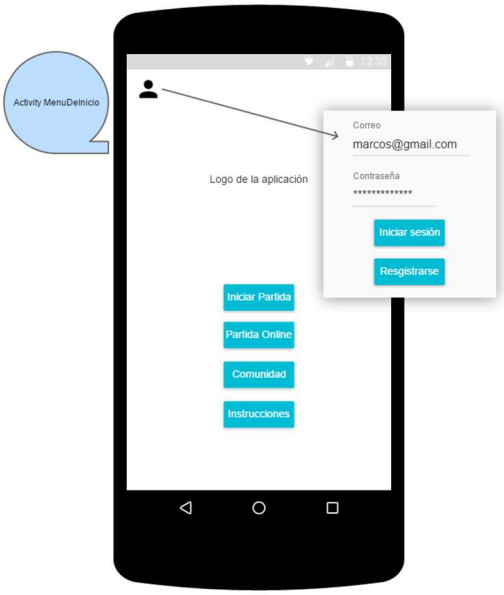


Figura: 17. Boceto menú principal

3.9.2 Modos de juego online y offline

Ambos modos de juego tendrán interfaces muy parecidas. Al principio aparecerá una vista para introducir valores iniciales de la partida, así como los nombres de los participantes. También estará el botón de empezar partida.

La siguiente vista corresponde al escenario, donde aparece el temporizador, eventos y un ranking con la puntuación actualizada de los jugadores. Para iniciar el temporizador también se ofrecerá un botón, y los eventos se mostrarán en ventanas emergentes superpuestas a la vista principal. Estos eventos podrán ser quitados pulsando fuera del diálogo o en el botón cancelar correspondiente.

La vista de introducir datos estará compuesta por dos secciones: valores globales y valores personales de cada jugador. En el primer caso se introducen valores numéricos de recursos financieros en común, y en el segundo cada jugador introduce cantidades de los productos en posesión. Habrá un botón para confirmar los datos

escritos, y al pulsarlo aparecerá la ventana de aviso por si hay algún dato erróneo.

Por último, la vista de resumen de la partida, en la que se podrá ver gráficas, volver al menú principal, y subir a la comunidad la partida jugada. A continuación, se muestran unos bocetos de estas interfaces:

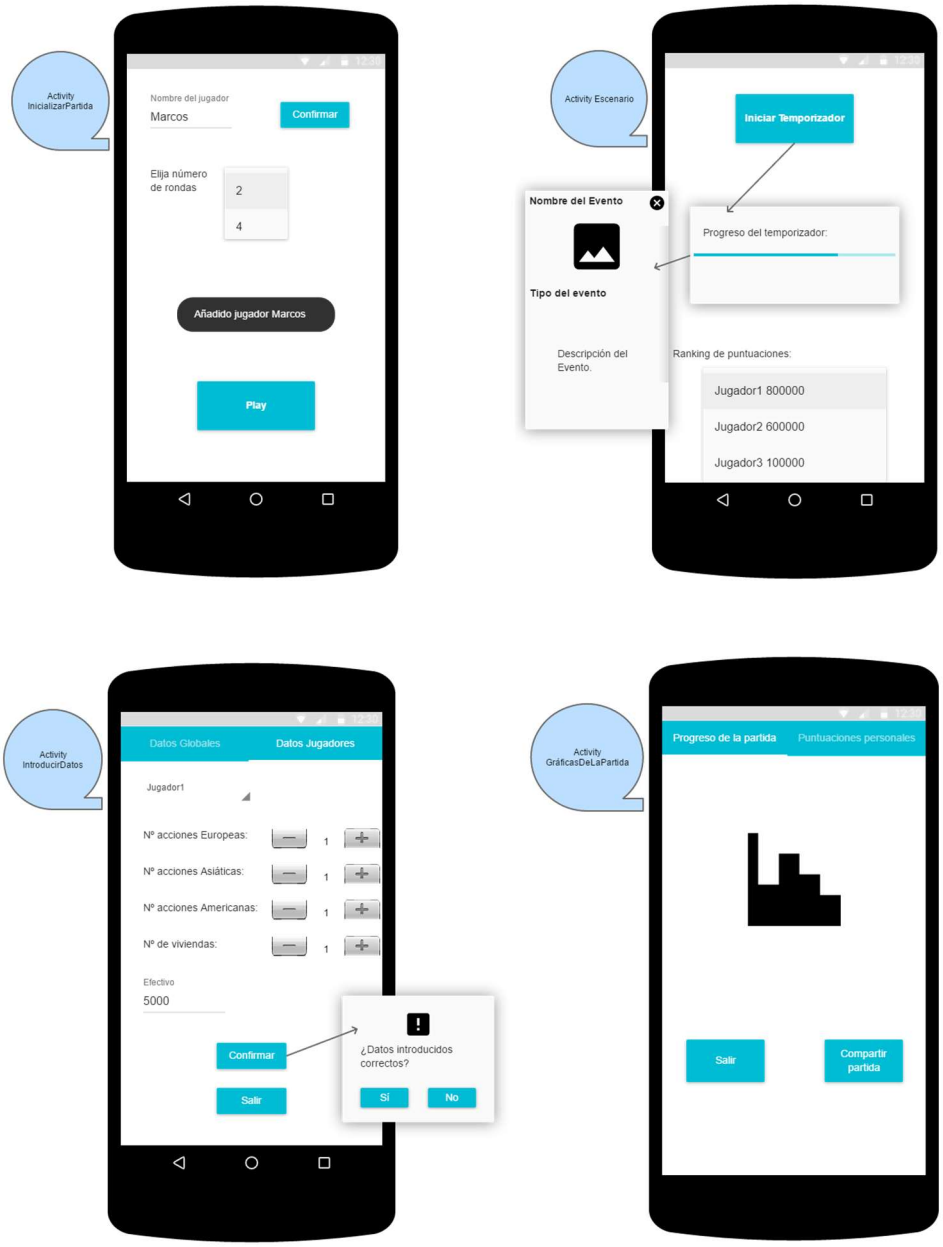


Figura: 18. Bocetos partida offline y online

3.9.3 Gestión de la comunidad

Esta interfaz sólo incluye una vista, y varias ventanas (fragments), que irán variando según la opción que el usuario pulse en un menú. En ella podrá gestionar su perfil, ver el ranking online, ver sus partidas subidas, ver la semilla distribuida disponible y los logros obtenidos. A continuación, se muestra un boceto de esta interfaz:

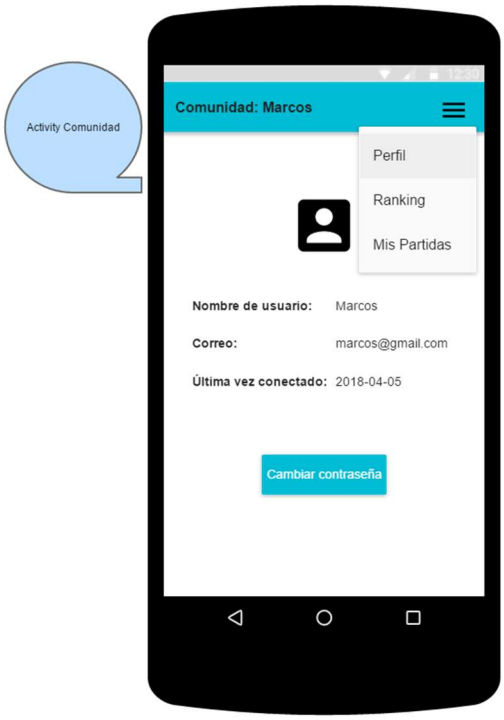


Figura: 19. Boceto vista comunidad

3.9.4 Visualización de manuales

Esta interfaz estará compuesta por una sola vista, donde se mostrará distintos documentos en los cuales los jugadores podrán aprender cómo jugar y cómo utilizar la aplicación. Estos documentos serán archivos pdf embebidos en el programa. A continuación, se muestra un boceto de esta interfaz:

Figura: 20. Boceto visualización de manuales

4 DISEÑO DEL SISTEMA

A partir de la Especificación de Requisitos del sistema, el siguiente paso es diseñar la aplicación. En este capítulo explicaremos, de forma estructurada y con la ayuda de diagramas UML, los distintos subsistemas del proyecto. El diseño se ha organizado por casos de uso.

Empezaremos desde un punto de vista más general, veremos cómo gestiona la aplicación el acceso a los distintos módulos (gracias a la pantalla del menú principal), y continuaremos reduciendo el nivel de abstracción para diseñar más detenidamente cada subsistema.

Debemos resaltar que se ha utilizado en el diseño los patrones Modelo Vista Controlador y Singleton, como se muestra en los diagramas. En el proyecto Android, la vista estará compuesta por todos los ficheros que representan el layout de la aplicación (son archivos xml), los controladores son todas las clases que gestionan los eventos e interacciones del usuario con el programa (actividades y fragmentos), además de las clases que gestionan las peticiones al servidor y procesar las respuestas. Finalmente, el modelo está compuesto por las clases que definen los datos y aquellas que trabajan directamente con la BBDD local (realizan sentencias sql).

Todos los servicios y la clase VolleyApplication siguen el patrón Singleton. Para facilitar la lectura de los diagramas, sólo se mostrará en el primer diagrama de secuencia el intercambio de mensajes para obtener esa instancia, ya que el procedimiento es exactamente el mismo en todos los subsistemas.

Además, utilizaremos el patrón Callback para procesar los datos recibidos del servidor, y actualizar dinámicamente la información mostrada en pantalla.

Por último, debemos recordar que para poder acceder al servicio web (excepto identificar usuario y registrar), debemos identificar el cliente con el token ClaveAPI, almacenado localmente de forma persistente.

4.1 Diseño global

4.1.1 Diagrama de paquetes

En este diagrama podemos ver cómo se divide el sistema, cumpliendo la especificación de requisitos.

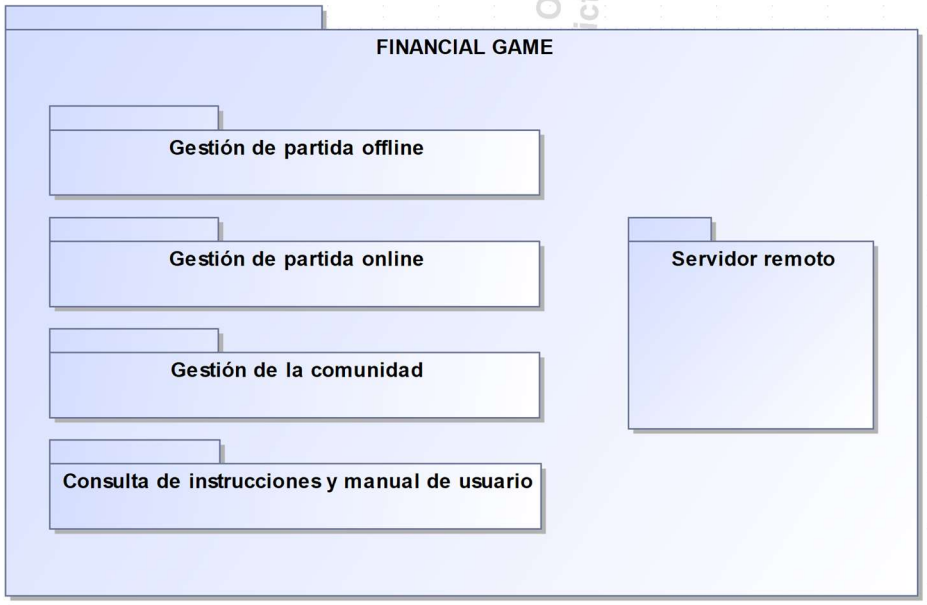


Figura: 21. Diagrama de paquetes del sistema

4.1.1 Diagrama de componentes

En esta imagen, hemos querido representar las distintas clases disponibles en la aplicación Android que gestionan las peticiones a servicios web alojados en el servidor. Para ello utilizan la clase *VolleyApplication*, que gestiona las peticiones HTTP en una cola y las manda. También atiende las respuestas del servidor. Como sabemos, todos los subsistemas que tengan funcionalidad online deberán conseguir la claveAPI. Por eso la clase *AlmacenSessionImpl* ofrece su interfaz con el método que devuelve esa información.

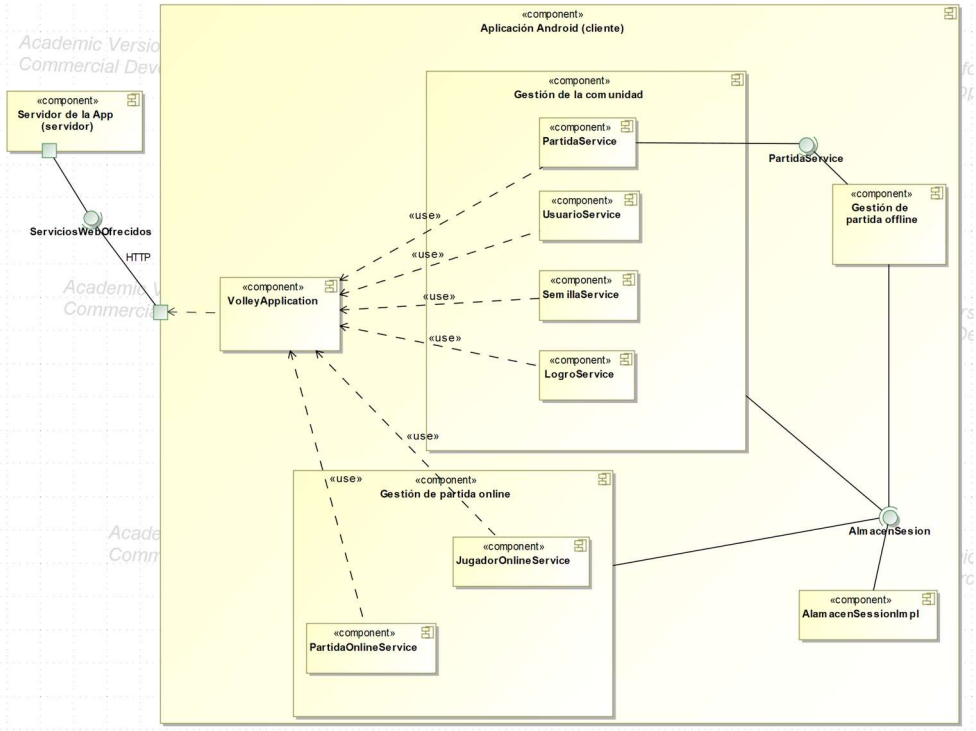


Figura: 22. Diagrama de componentes, servicios web

La interfaz REST del api correspondiente a los servicios web, la hemos denominado *ServiciosWebOfrecidos*. A continuación, se muestra una tabla de esa interfaz, identificando los recursos, url, métodos HTTP y parámetros necesarios para los mensajes POST y UPDATE.

Recurso	URL	Método HTTP	Parámetros
Registrar usuario	https://website/v1/usuario/registro	POST	{nombre, correo, contrasenia}
Autenticar usuario	https://website/v1/usuario/login	POST	{correo, contrasenia}
Cambiar contraseña	https://website/v1/usuario/cambiarContrasenia	UPDATE	{contrasenia nueva}
Partidas subidas	https://website/v1/partida/partidasSubidas	GET	-
Ranking	https://website/v1/partida/ranking	GET	-

Partida	https://website/v1/partida/#id	POST GET DELETE	{fecha, numrondas, nombreganador, numjugadores, puntuacionganador}
Actualizar fecha conexión	https://website/v1/usuario/actualizarConexion	UPDATE	{fecha}
Logros	https://website/v1/logro/obtenerLogros	GET	-
Logro	https://website/v1/logro/#id	GET	-
Semillas	https://website/v1/semilla/obtenerSemillas	GET	-
Partidas Online	https://website/v1/partidaonline/partidasOnline	GET	-
Partida online	https://website/v1/partidaonline/crearPartidaOnline/#id	POST GET DELETE	{nombreanfitrion, fecha, numrondas, numjugadores max}
Jugador online	https://website/v1/jugadoronline/jugadorOnline/#id	POST GET DELETE	{estado, isanfitrion, idPartidaOnline}
Estado de la partida online	https://website/v1/partidaonline/estadoPartidaOnline/#id	UPDATE GET	{estado}
Jugadores online	https://website/v1/jugadoronline/jugadoresOnline	GET	-
Obtener identificadores online	https://website/v1/jugadoronline/idsOnline	GET	-
Estado del jugador online	https://website/v1/jugadoronline/estadoJugadorOnline/#id	GET UPDATE	{estado}

4.1.2 Diagramas de clases

En esta figura se representa el modelo de datos de la aplicación: atributos, métodos y relaciones entre ellos.

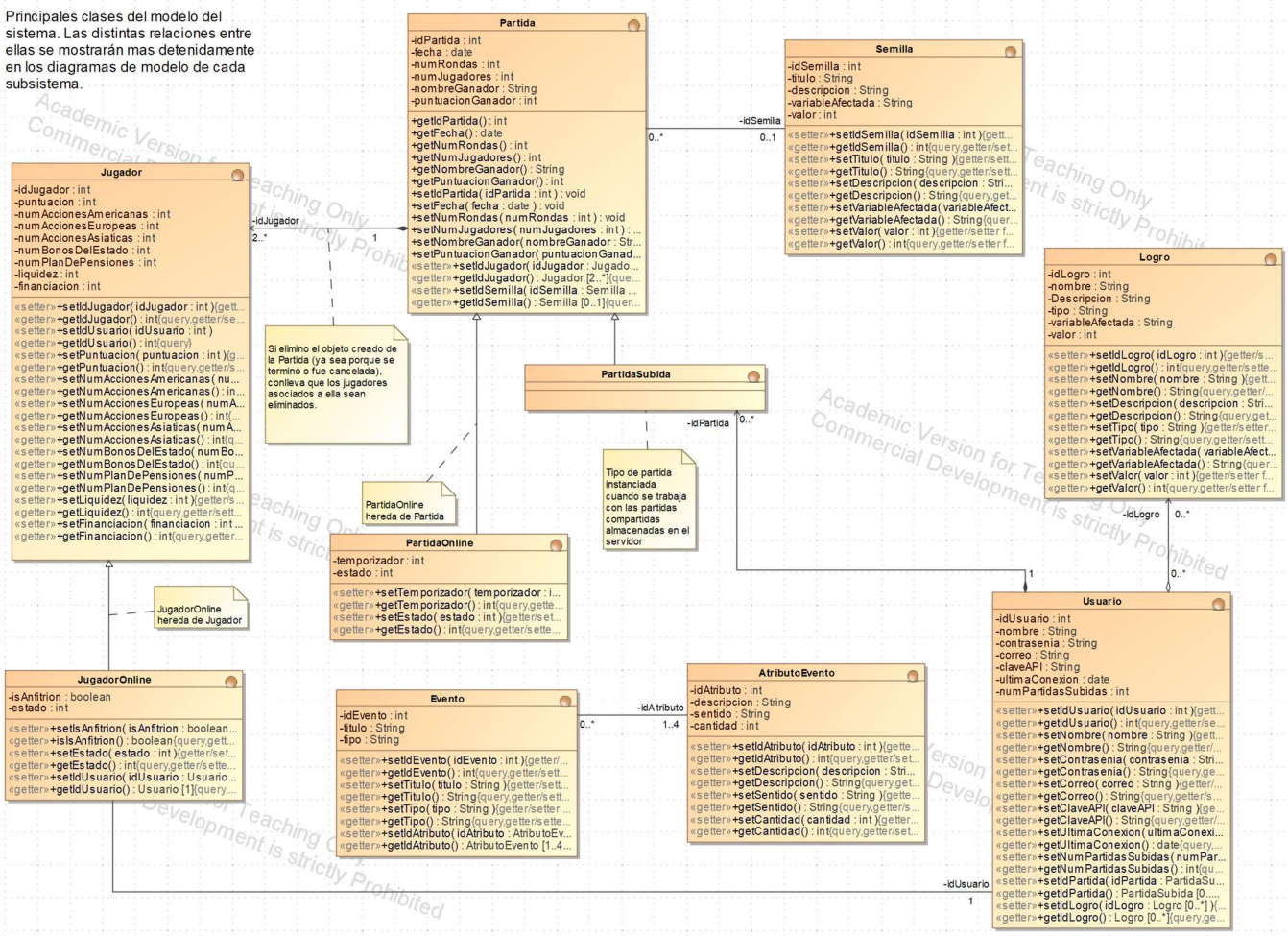


Figura: 23. Diagrama de clases del sistema

Debido a las peculiaridades de este proyecto (es una aplicación Android) es necesario especificar clases específicas de este tipo de desarrollos. Una actividad (Activity) es el controlador de la pantalla que visualiza el usuario en su dispositivo, mientras que un fragmento (Fragment) ocupa sólo una parte de ésta (como por ejemplo diálogos y ventanas emergentes). En la siguiente figura podemos ver todas las clases que heredan de este tipo de clases.

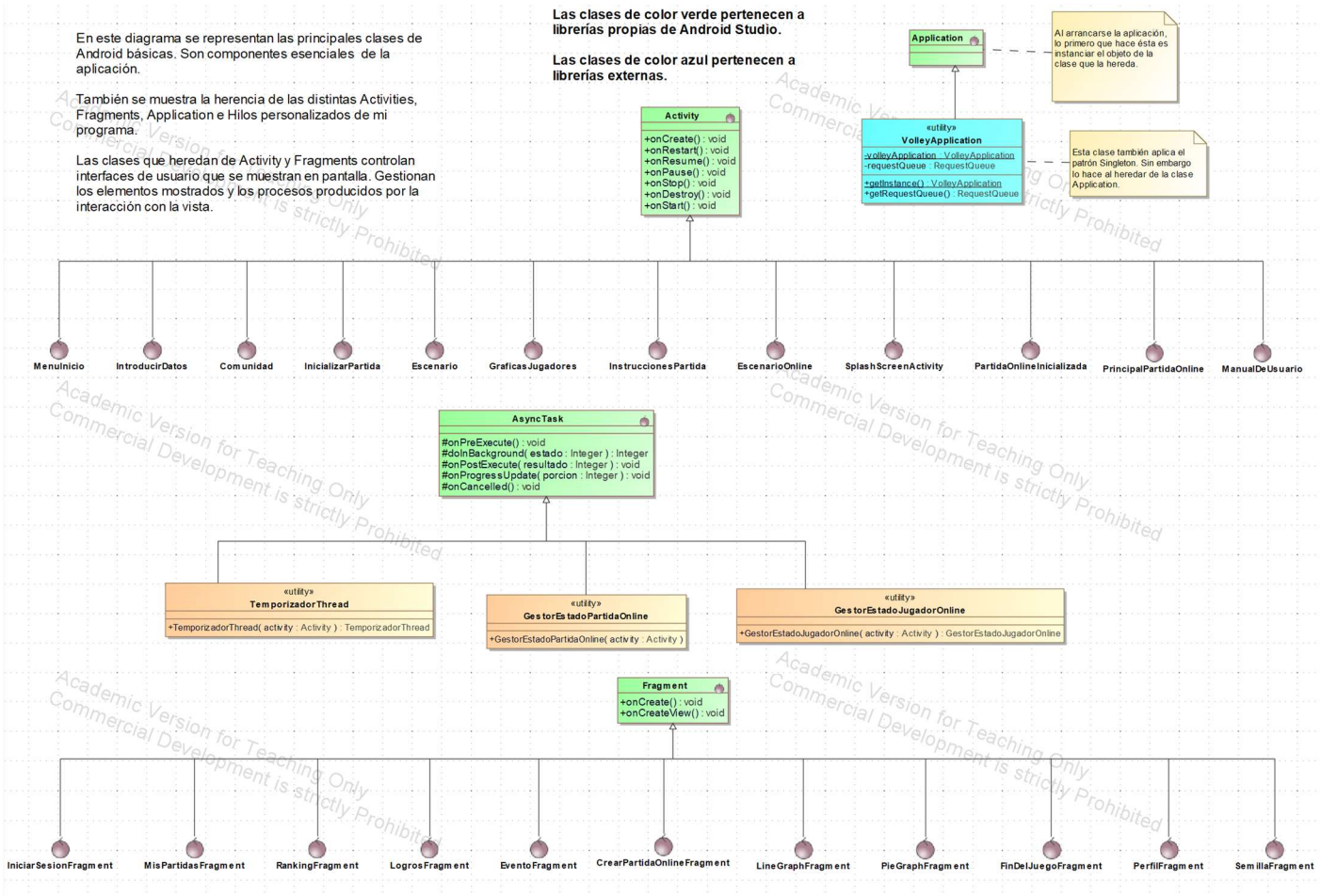


Figura: 24. Diagrama de clases del sistema, herencias

Aquellas clases con el estereotipo *utility* no gestionan elementos del modelo o de la vista, únicamente trabajan con otras clases del estereotipo controlador. Por lo tanto, no podemos catalogarla dentro del patrón MVC. Realizarán tareas varias como creación de subprocesos e hilos, o gestionar peticiones y respuestas HTTP.

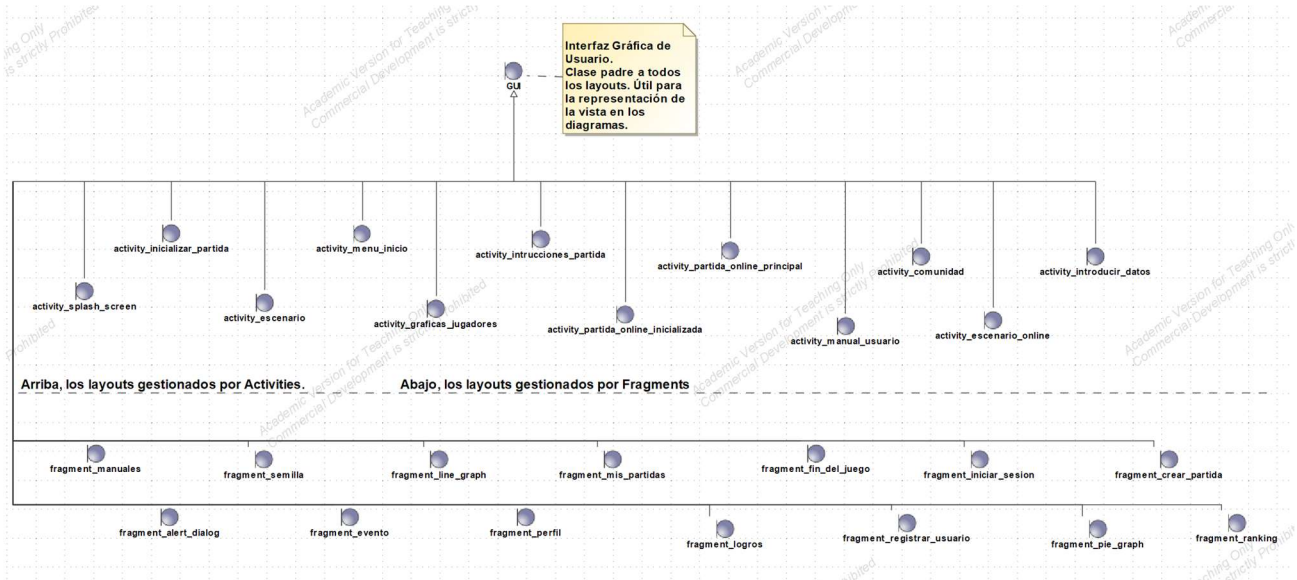


Figura: 25. Diagrama de clases del sistema, vista

En la figura superior, se muestran una serie de clases que implementan el estereotipo vista del patrón MVC. Representan la interfaz gráfica de la aplicación, la cual interactúa directamente un usuario a través de la pantalla de su dispositivo. En el desarrollo Android, la vista se programa en archivos xml, haciendo uso de etiquetas para añadir componentes a esa interfaz. Cada actividad y fragmento gestionan su propio layout.

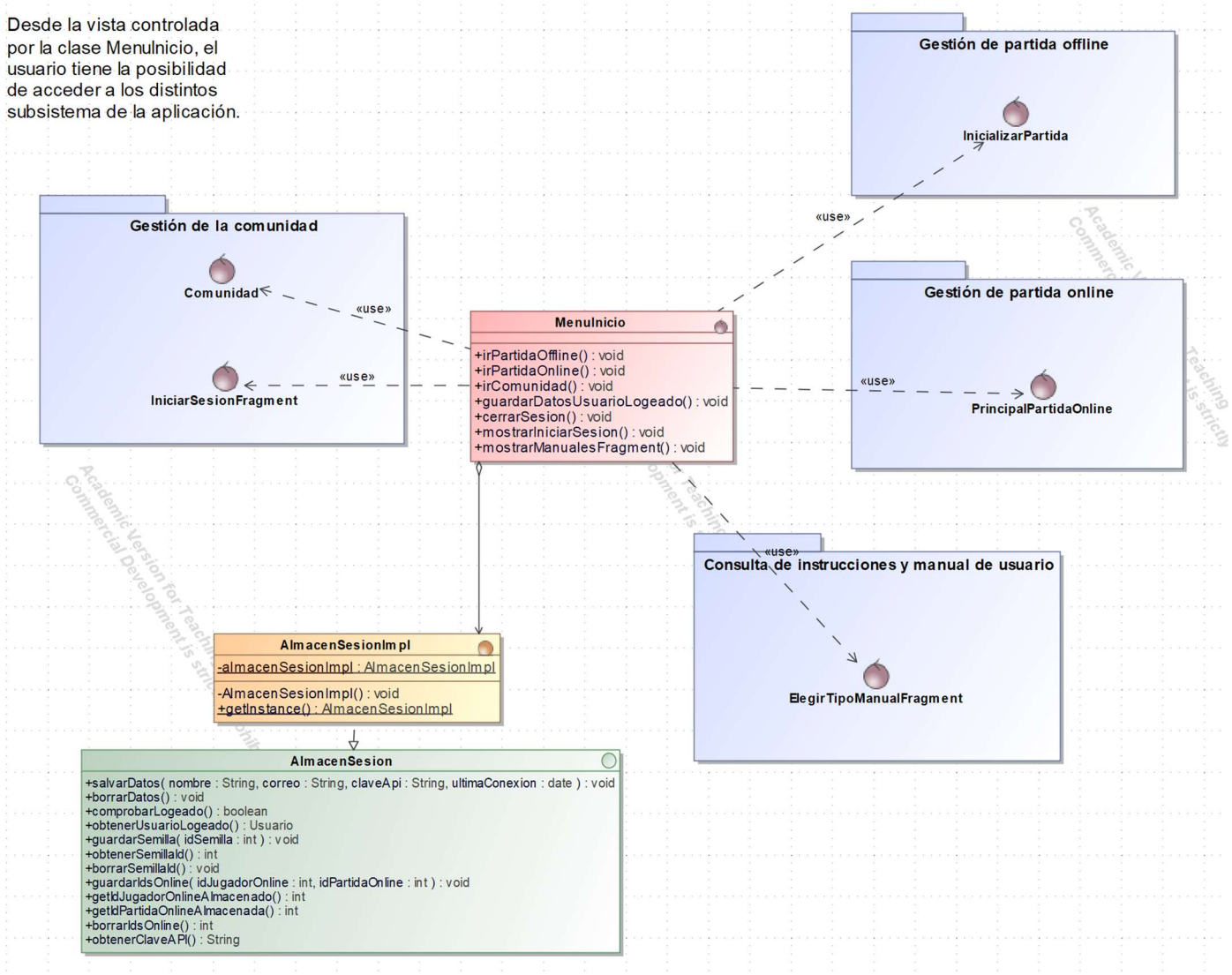


Figura: 26. Diagrama de clases del sistema, pantalla de menú principal

En este diagrama se representa las relaciones entre las distintas clases que intervienen en la pantalla de inicio. Recordemos que, desde esa sección, el usuario puede acceder a los distintos subsistemas.

Por características del desarrollo Android, una actividad lanza a otra, y si es eliminada se muestra la actividad padre. La clase *AlmacenSessionImpl* pertenece al modelo de la aplicación ya que trabaja directamente con la BBDD local, almacenando información importante relacionada con el usuario identificado.

4.1.3 Diagramas de actividad

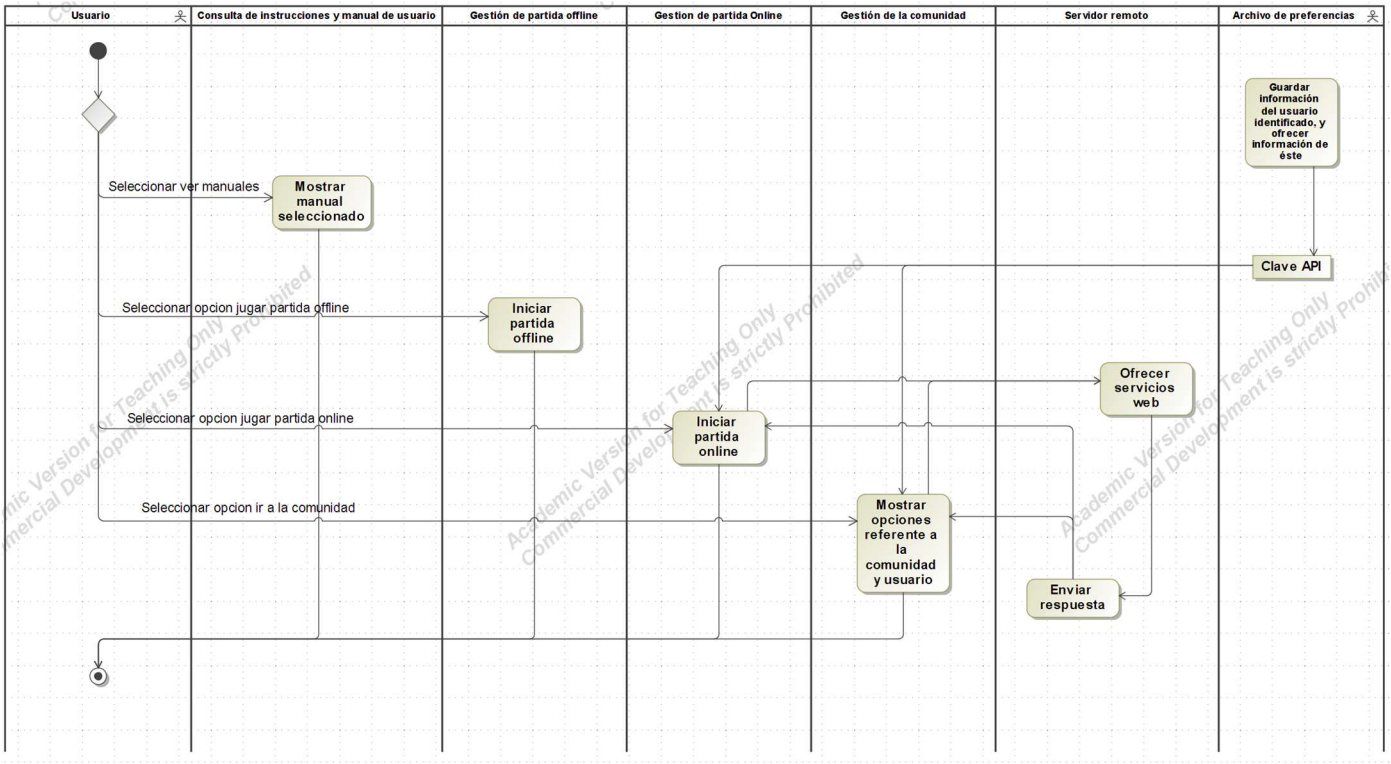


Figura: 27. Diagrama de actividad del sistema

En esta figura se muestra la secuencia de acciones cuando una persona quiere acceder a las distintas opciones principales de la aplicación. Recordemos que para poder entrar en *Gestión de la comunidad* y *Partida online* debemos estar identificados, y por tanto tener almacenada la claveAPI en nuestro archivo de preferencias.

4.1.4 Diagramas de secuencia

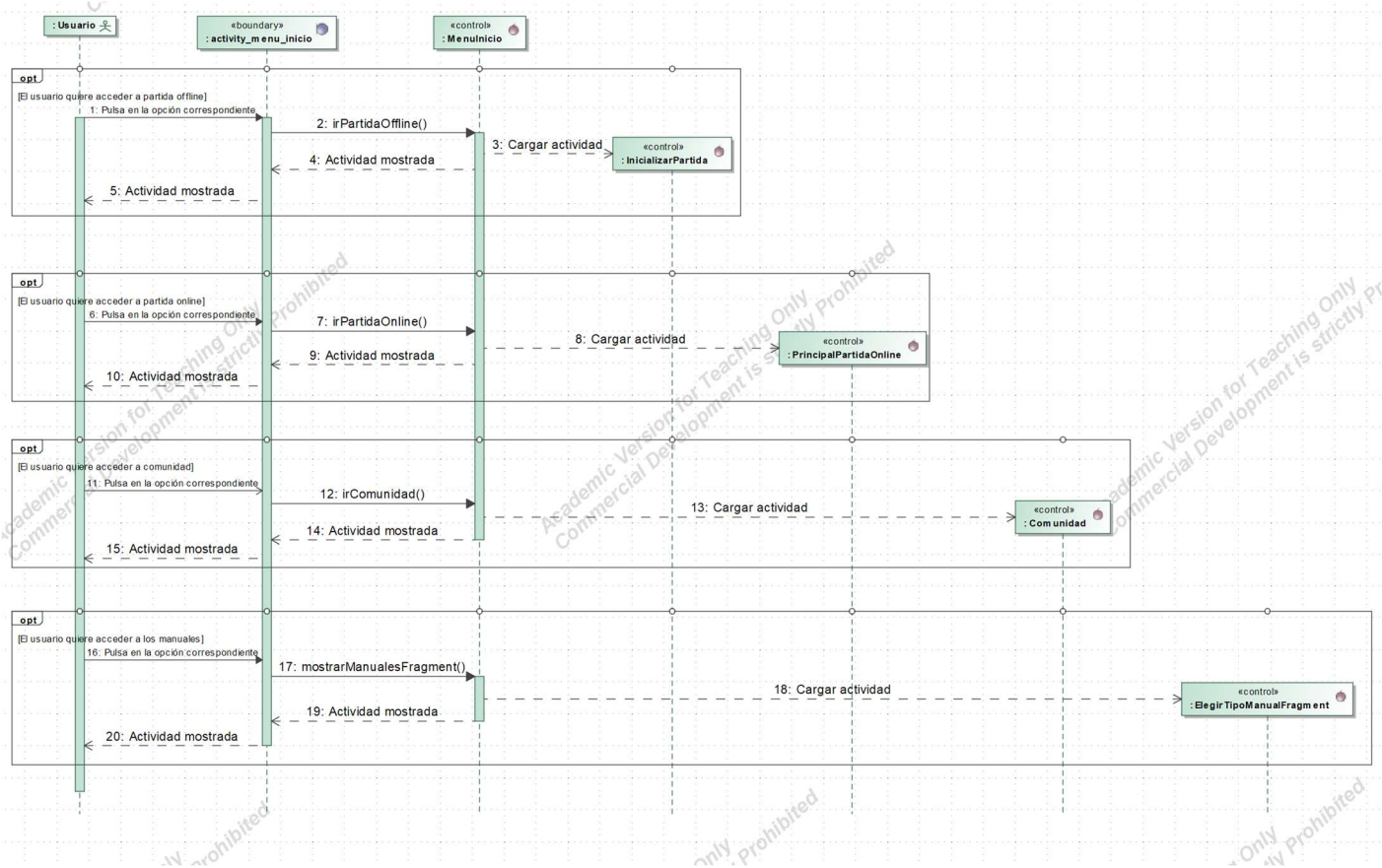


Figura: 28. Diagrama de secuencias del sistema

El comportamiento de una actividad es característico, ya que cuando se instancia un objeto de ella se muestra en pantalla, pasando la actividad padre (quien lanza la nueva) a una pila. En esta imagen podemos ver el intercambio de mensajes cuando alguien interactúa con el menú principal de la aplicación, diferenciando cada opción disponible, que representa un caso de uso.

4.1.5 Modelo de la BBDD remota

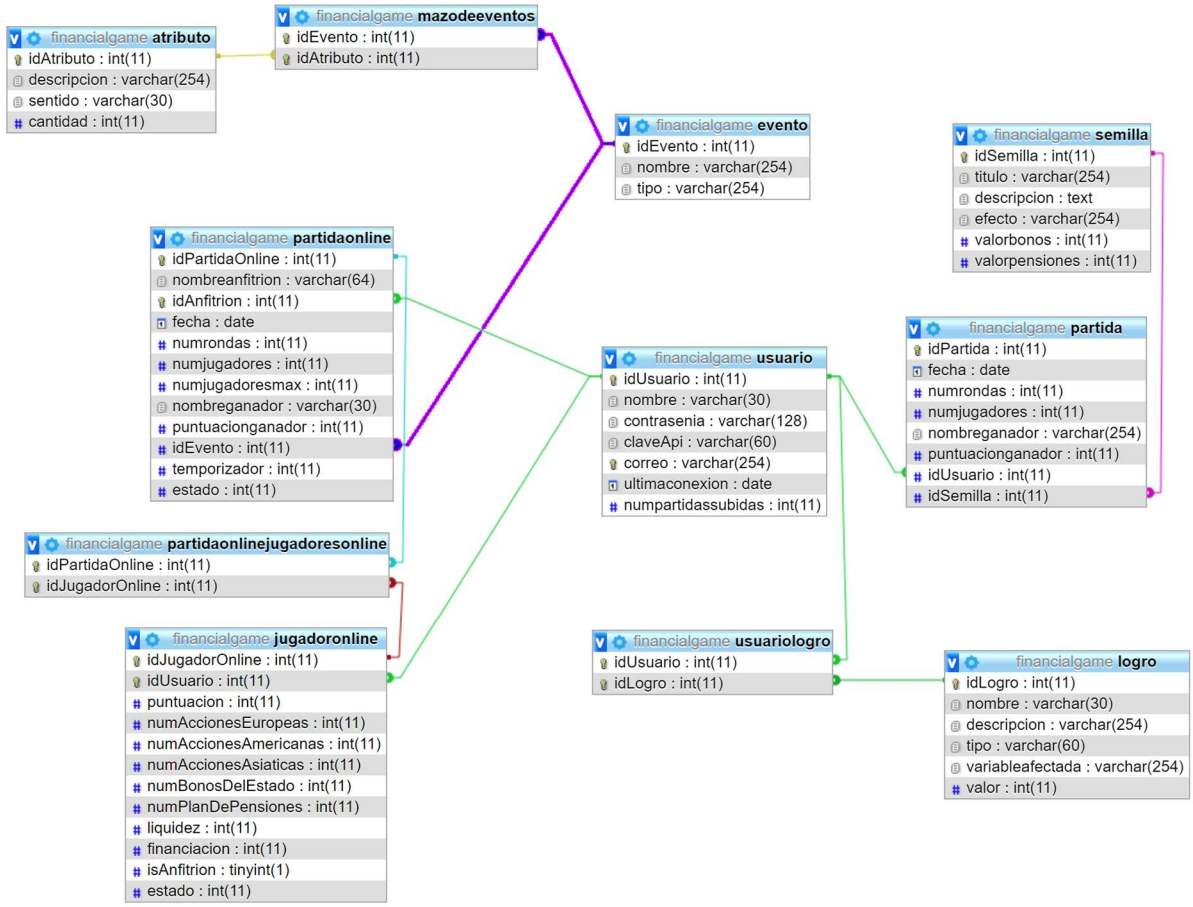


Figura: 29. Modelo de la BBDD remota

Esta imagen corresponde al modela de la BBDD remota, la cual tiene acceso el servidor. Podemos ver los atributos, las *primaryKeys* y las *foreignKeys*. Es muy interesante comprobar que tiene un gran parecido con el diseño del modelo del sistema. Esto es así porque para mantener la integridad de la información cuando enviamos y recibimos datos en las peticiones http, haciendo uso del paradigma REST (orientado a recursos, que son los mostrados en el diagrama).

4.3 Diseño por casos de uso

4.3.1 Gestión de partida offline

4.1.5.1 Diagramas de clases

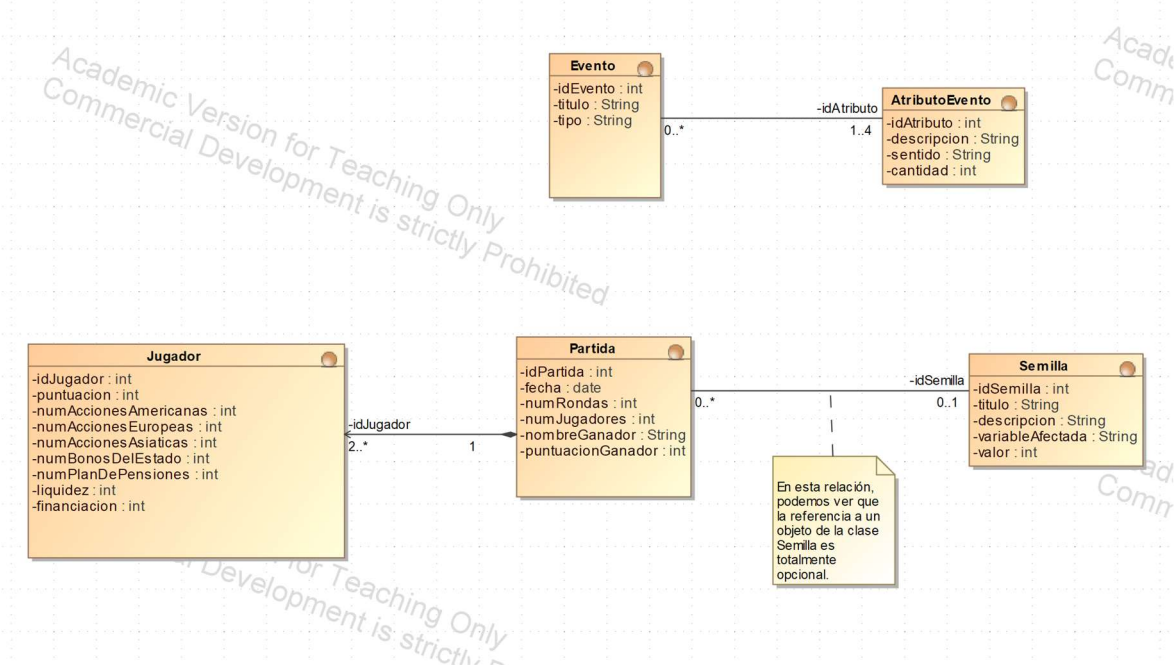


Figura: 30. Diagrama de clases del modelo partida offline

A partir de esta imagen podemos apreciar que un evento no tiene que tener obligatoriamente atributos asociados. Lo mismo pasa entre *Partida* y *Semilla*. Sin embargo, sí es esencial que una partida tenga al menos un jugador asignado para poder existir. Un jugador sólo puede estar en una partida.

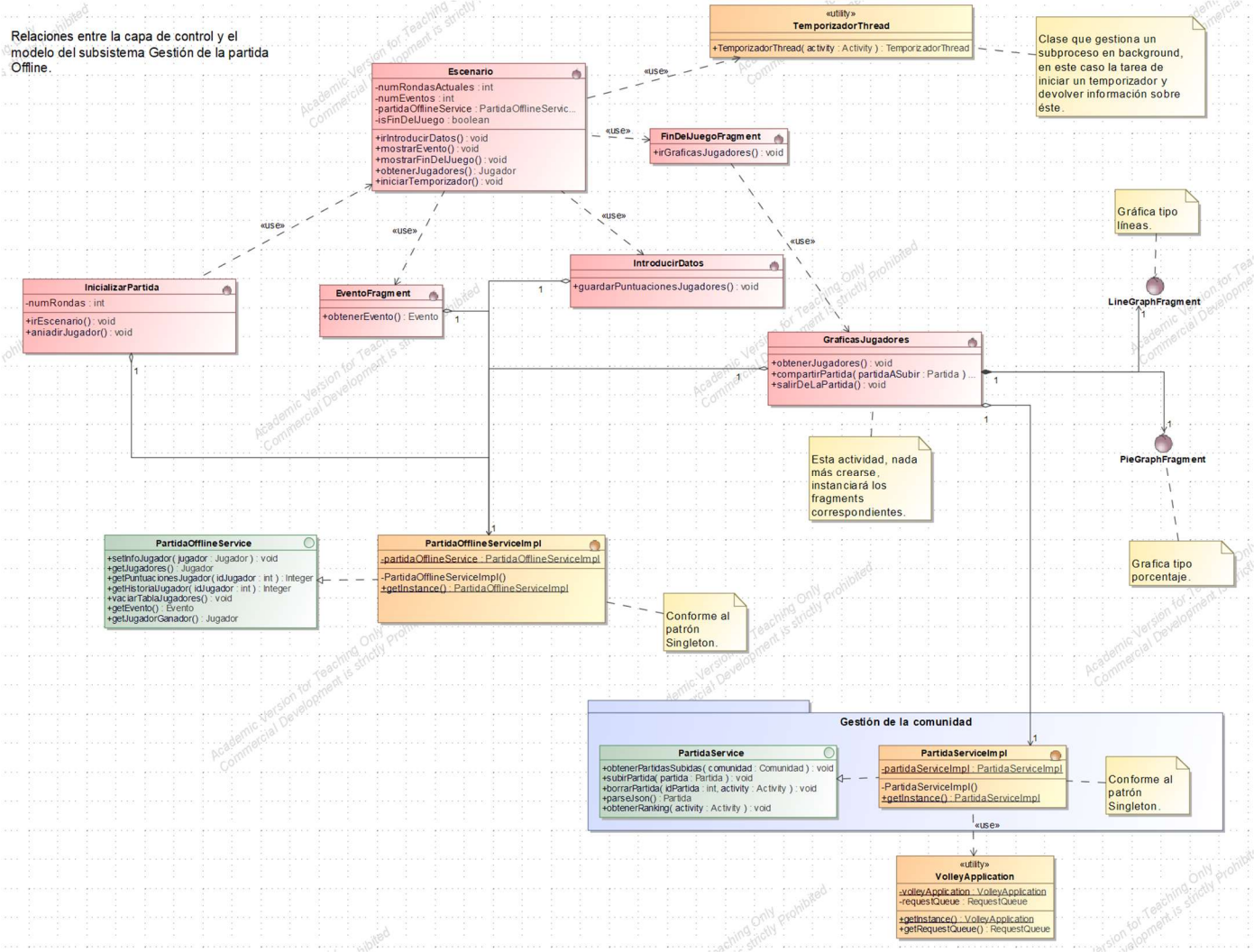


Figura: 31. Diagrama de clases de partida offline

En esta figura se representan todas las clases que participan en el susbistema, y cómo se relacionan entre ellas. Podemos apreciar que este módulo puede acceder a un servicio perteneciente al módulo comunidad, ya que como se especifica en el documento ERS el jugador podrá subir la partida offline al servidor. *GraficasJugadores* tiene una relación fuerte con las dos clases que gestionan las gráficas debido a que son inmediatamente lanzadas cuando se carga la actividad, no son separables y comparten el mismo ciclo de vida.

4.1.5.2 Diagramas de objetos

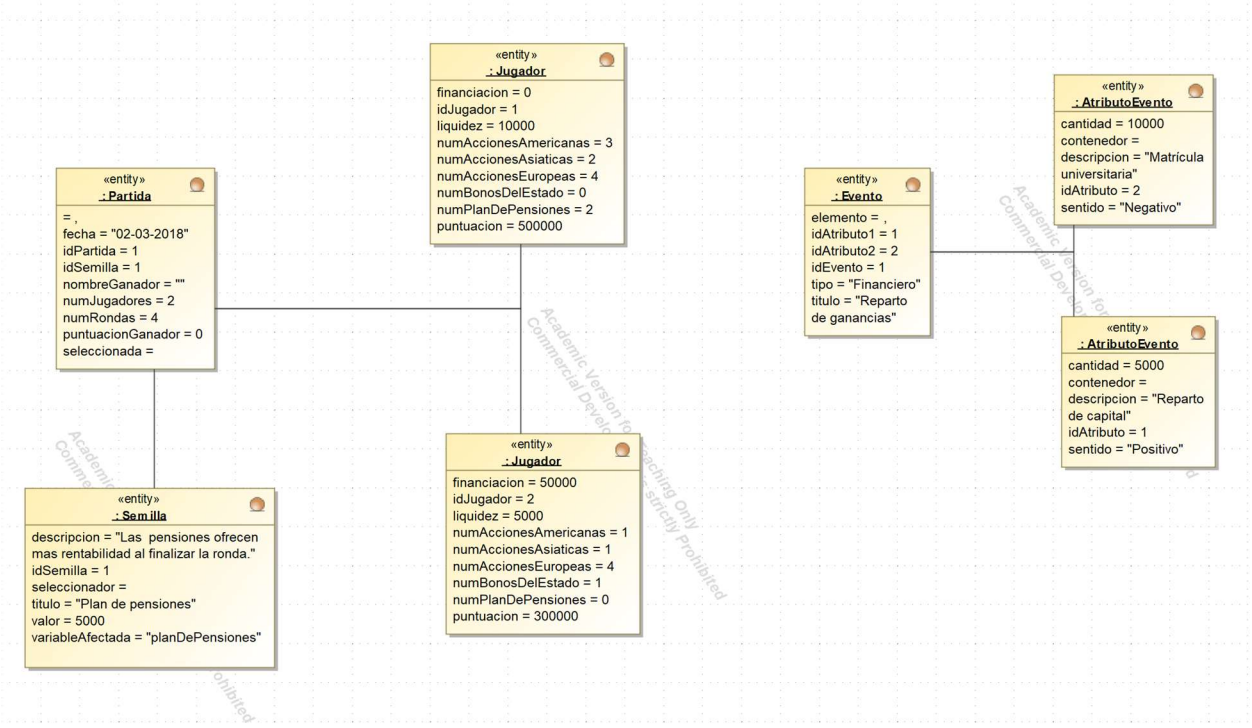


Figura: 32. Diagrama de objetos de partida offline en proceso

En el siguiente diagrama muestra un momento concreto de una partida que está en proceso, con dos jugadores en la partida y sus respectivas puntuaciones. También hay activa una semilla, que afectará en las variables de la partida (en este caso *planDePensiones*). También tenemos un evento instanciado, con dos atributos.

4.1.5.3 Diagramas de actividad

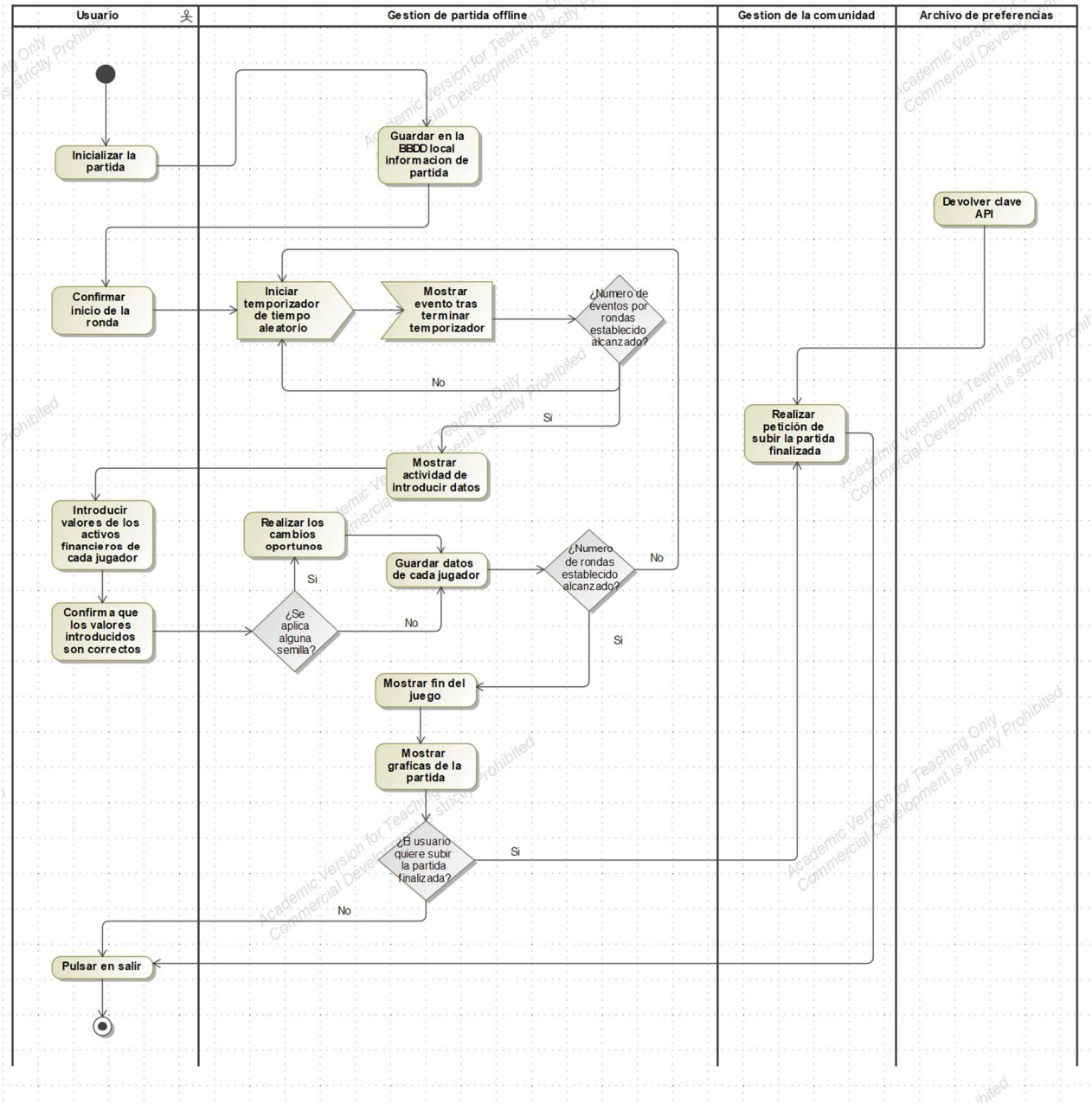


Figura: 33. Diagrama de actividad de partida offline

En este diagrama se muestra la secuencia ordenada de las distintas actividades del sistema cuando el usuario entra en el módulo *Partida offline* y juega la partida. Podemos ver que cuando se inicia el temporizador, no pasamos a la siguiente actividad hasta que se haya acabado el tiempo, pero como se realiza en otro subproceso, el subsistema espera una señal para pasar al siguiente paso (lo que hace el subproceso es llamar a un método del proceso original).

4.1.5.4 Diagramas de secuencia

A continuación, se representan los diagramas de secuencia de este módulo del sistema. Podemos ver claramente las clases organizadas según su estereotipo del patrón MVC. Aunque en estas imágenes aparecen todas las clases de la vista equivalente a cada actividad y fragmento (los layouts), en los diagramas de secuencia de los siguientes apartados se simplificará para hacer más limpio el diseño.

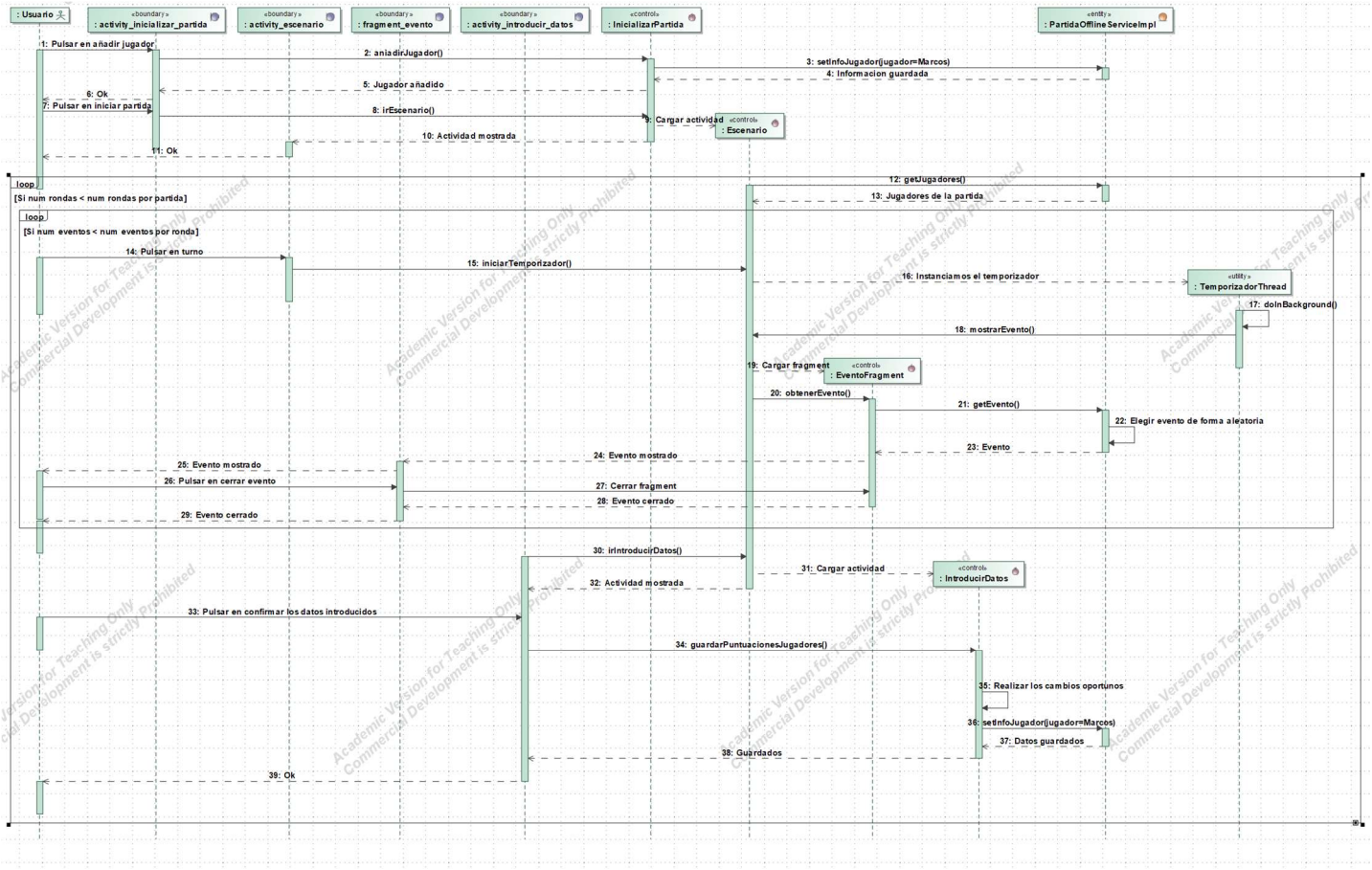


Figura: 34. Diagrama de secuencia de partida offline, primera parte

Esta primera parte abarca desde que alguien entra en este módulo hasta cuando se guardan las puntuaciones de los jugadores. El servicio utilizado para almacenar y obtener datos de la BBDD local será *PartidaServiceImpl*. Cuando de crea una instancia del temporizador, toda la carga principal de su comportamiento de aloja dentro de *doinBackground()*, que a su vez se ejecuta en un proceso aparte, evitando que la aplicación se bloquee. La clase *Escenario* le ofrece un método callback para avisar del progreso del temporizador.

Podemos ver dos bucles, el interno representa en transcurso de la serie de eventos que compone una ronda (variable del sistema), y el externo las rondas de la partida (variable modificable por el usuario).

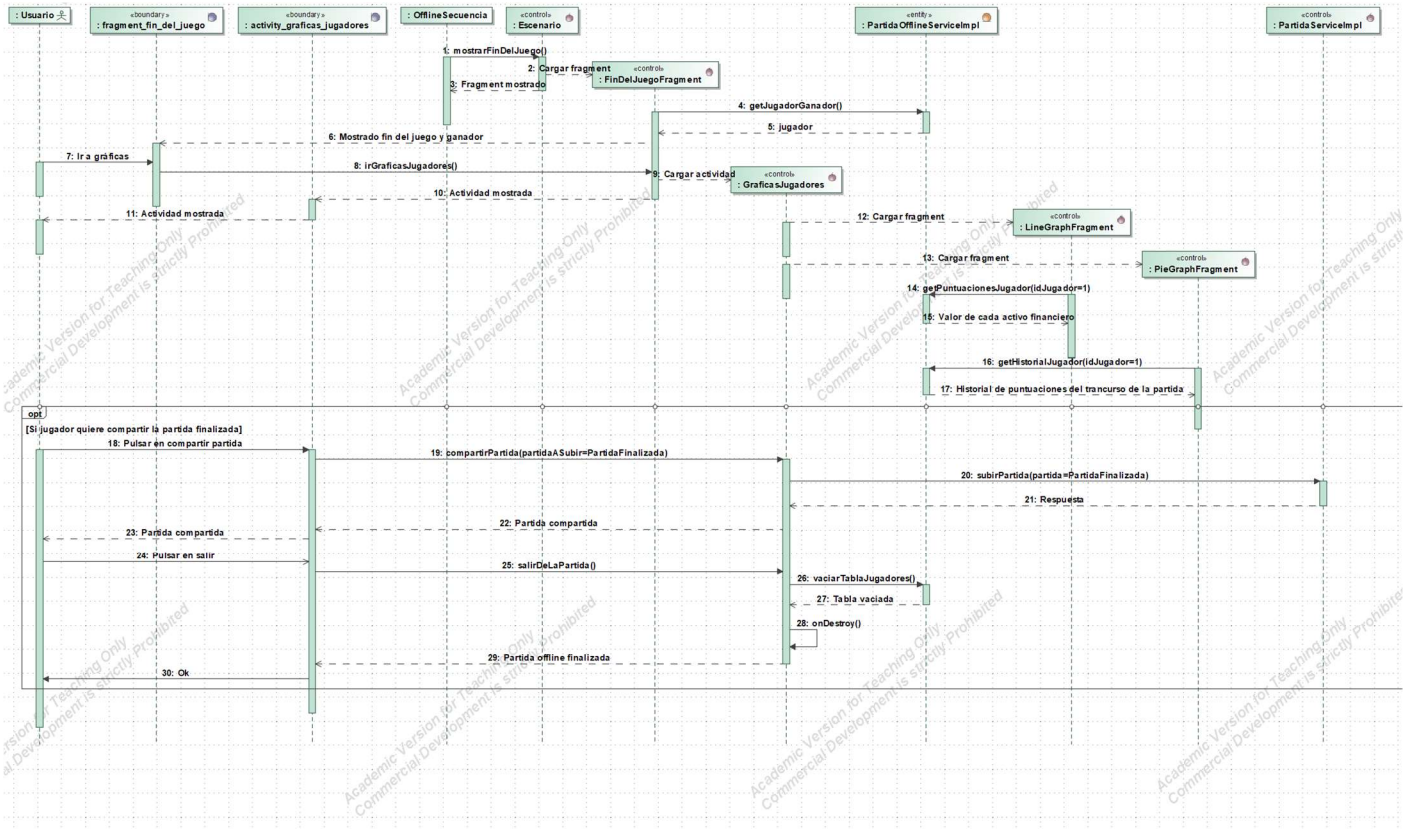


Figura: 35. Diagrama de secuencia de partida offline, segunda parte

Partiendo del diagrama anterior, en este se muestra el final de la partida. Si el jugador quiere compartir la partida, la actividad correspondiente se comunica con el servicio que manda la petición al servidor. Finalmente, cuando se sale de la partida, vaciamos los datos almacenados en la BBDD local y se ejecuta automáticamente *onDestroy()*.

4.3.2 Gestión de partida online

4.1.5.5 Diagramas de clase

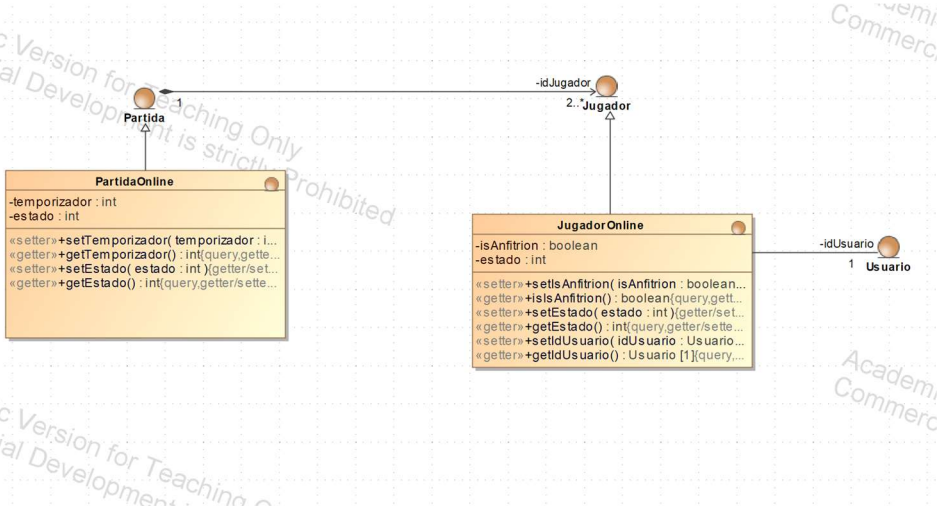


Figura: 36. Diagrama de clases del modelo de partida online

En esta imagen vemos el tipo de clases que se utilizarán para el modelo. Ambas son hijas de una clase padre debido a su gran similitud, de esta manera nos ahorramos tener que repetir atributos y comportamientos redundantes.

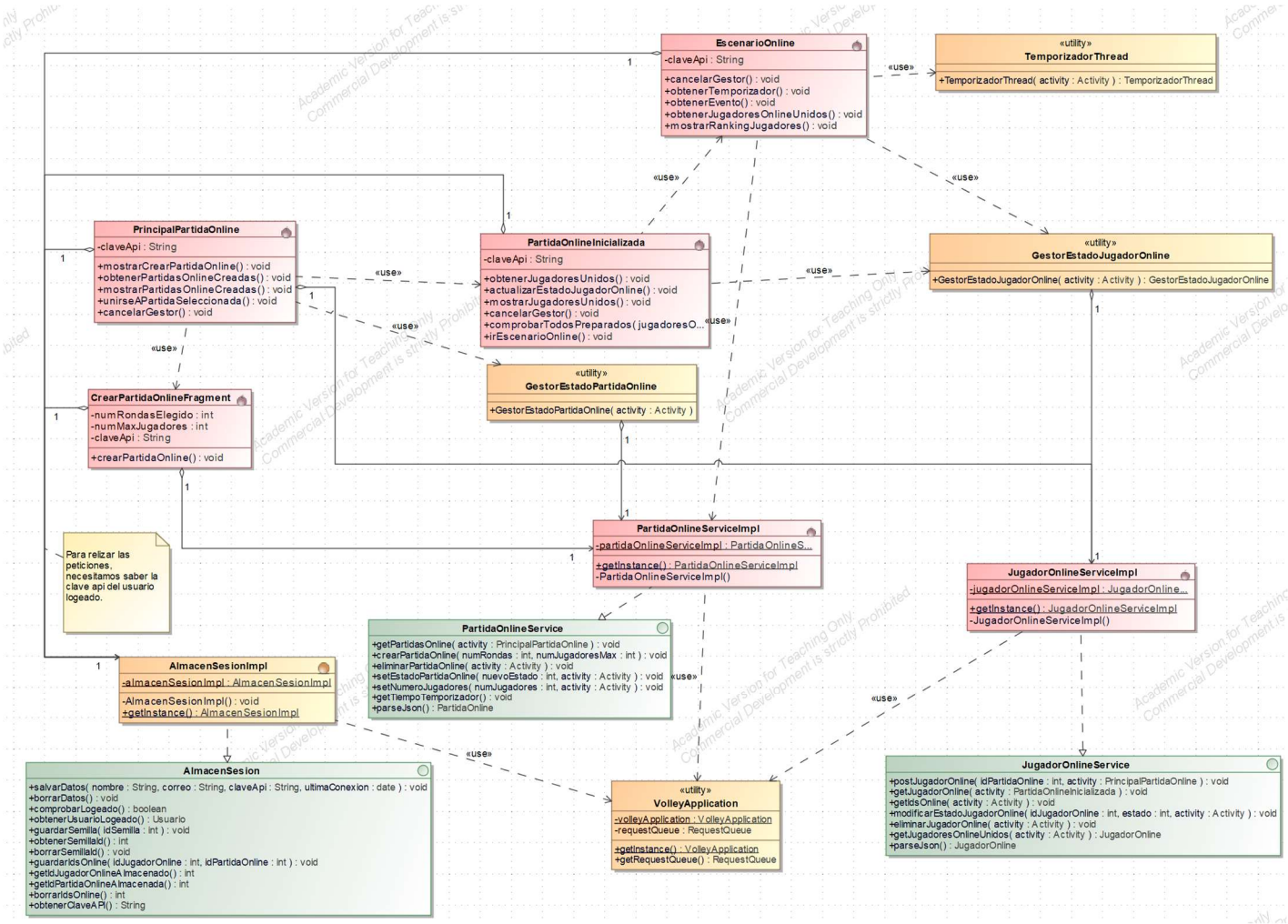


Figura: 37. Diagrama de clases de partida online

Como podemos apreciar, algunos servicios implementan el método llamado *parseJson()*. Como sabemos del ERS, la información transmitida en mensaje http entre cliente-servidor utiliza la estructura JSON. Debido a esto, los servicios necesitan una herramienta para decodificar la información y convertirla en un objeto de nuestro sistema. Gracias a que nuestro modelo respeta la arquitectura de las tablas de la BBDD remota, esta conversión es más sencilla (todas las columnas de la tabla están representadas como atributos en nuestro modelo).

4.1.5.6 Diagramas de objetos

Este diagrama muestra el momento concreto de una partida online iniciada, creada el 09-11-2018, con dos jugadores online participando, y uno de ellos (el de *idJugador* igual a 2) siendo el anfitrión.

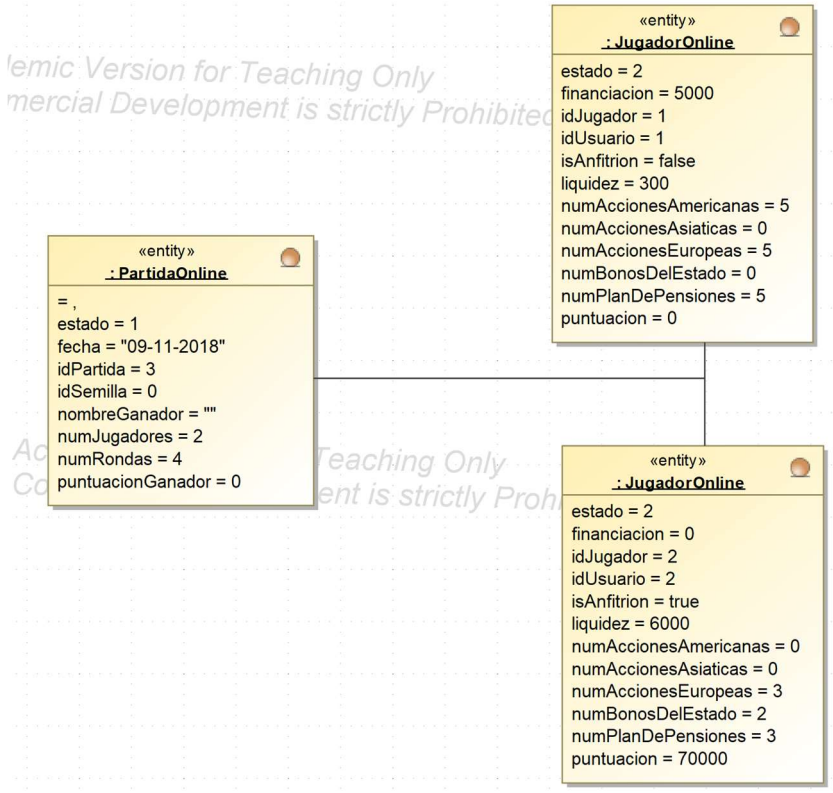


Figura: 38. Diagrama de objetos de partida online iniciada

4.1.5.7 Diagramas de actividad

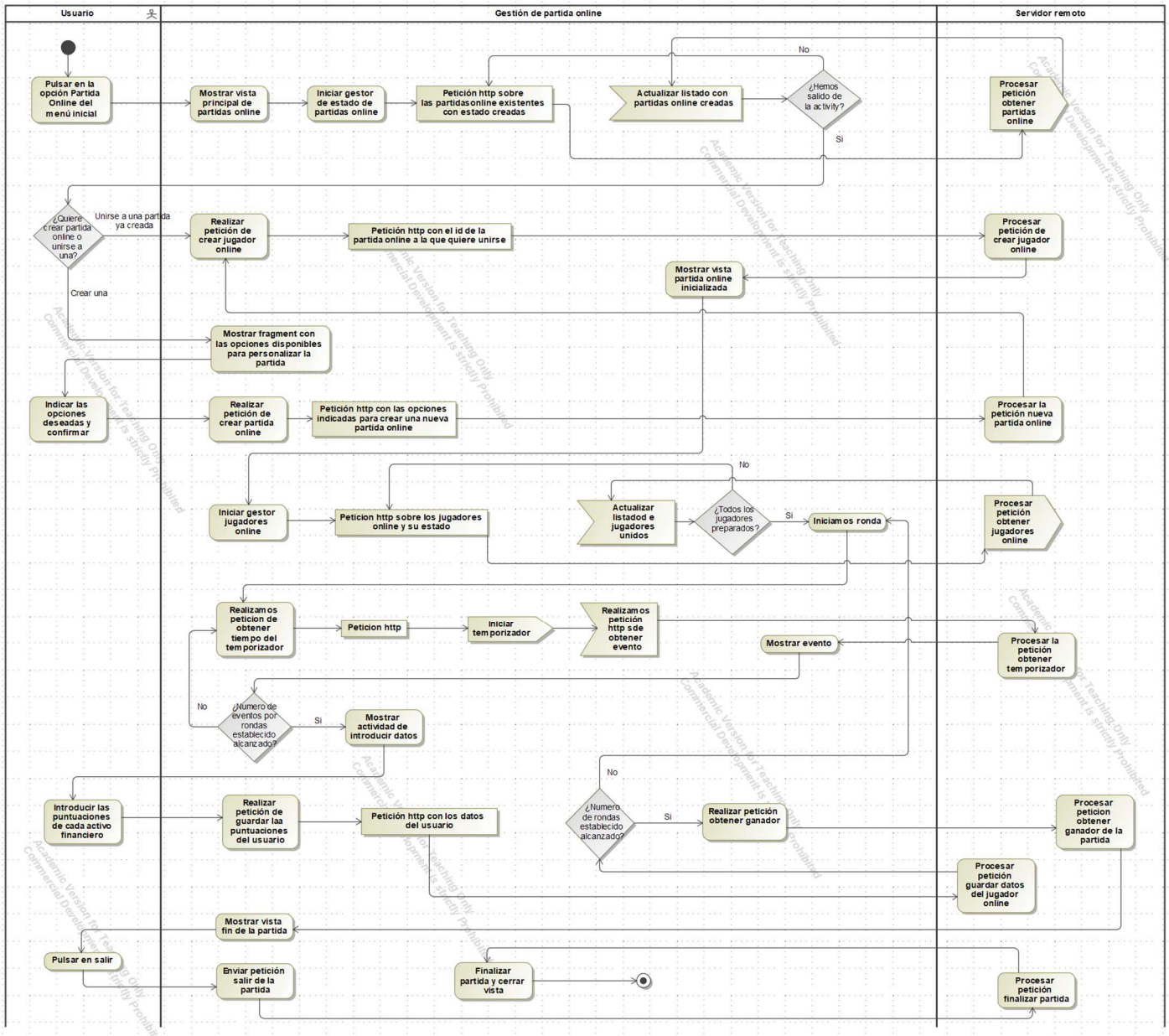


Figura: 39. Diagrama de actividad de partida online

En esta figura se muestra de principio a fin la serie de actividades que realiza el usuario con la aplicación (y las tareas internas del sistema) de una partida online. Cada vez que la aplicación realiza una actividad que conlleva enviar una petición al servidor, su siguiente actividad no se realiza hasta que no haya recibido un mensaje de respuesta, por lo tanto, se utiliza un método callback para gestionar esa respuesta y continuar con la secuencia de acciones. Esta peculiaridad se produce siempre que hacemos uso de un servicio web y queremos procesar la respuesta (en los diagramas de secuencia se verá con más detalle).

4.1.5.8 Diagramas de secuencia

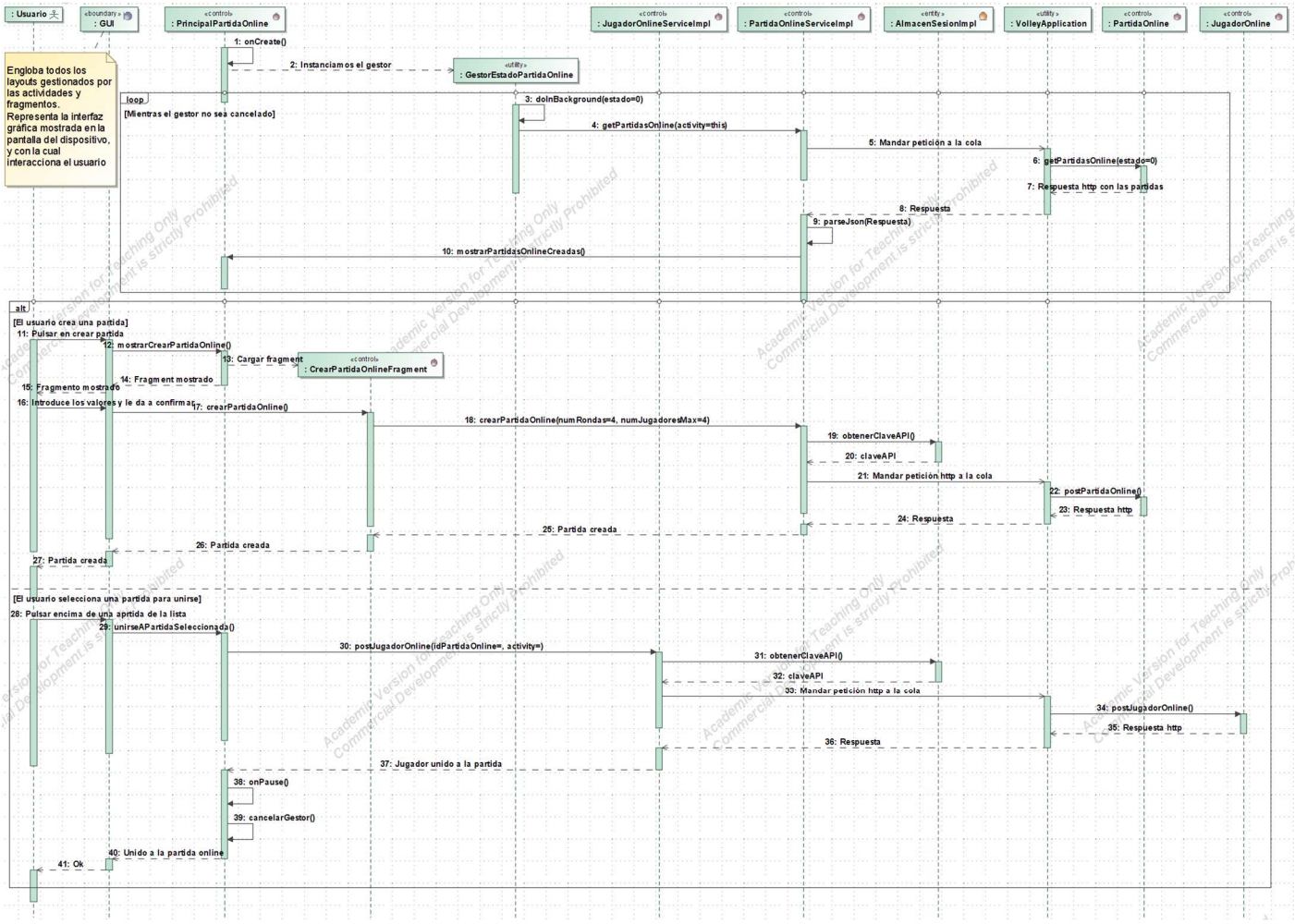


Figura: 40. Diagrama de secuencia de partida online, primera parte

Este diagrama muestra el intercambio de mensajes cuando un usuario entra en la actividad principal *Partida Online*, donde podrá crea una partida o unirse a una partida ya creada por otra persona.

Aquí entra en escena la clase *VolleyApplication*. Permite mandar peticiones asíncronas al servidor, pudiendo controlar cuando nos llegue la respuesta. Podemos apreciar que la llamada a esta clase se realiza con un mensaje asíncrono. En el servicio correspondiente, ya se le indica el método callback para recoger y gestionar la respuesta desde el controlador.

Por último, explicamos cómo funcionan los gestores de estado. Como vemos, es un subproceso que se ejecuta indefinidamente, y en otro proceso distinto al principal. Éste manda peticiones periódicas al servidor preguntando por los jugadores online (o partidas online) que tienen el estado indicado. El gestor se eliminará cuando no esté en primer plano la actividad que lo llama, o si es eliminada.

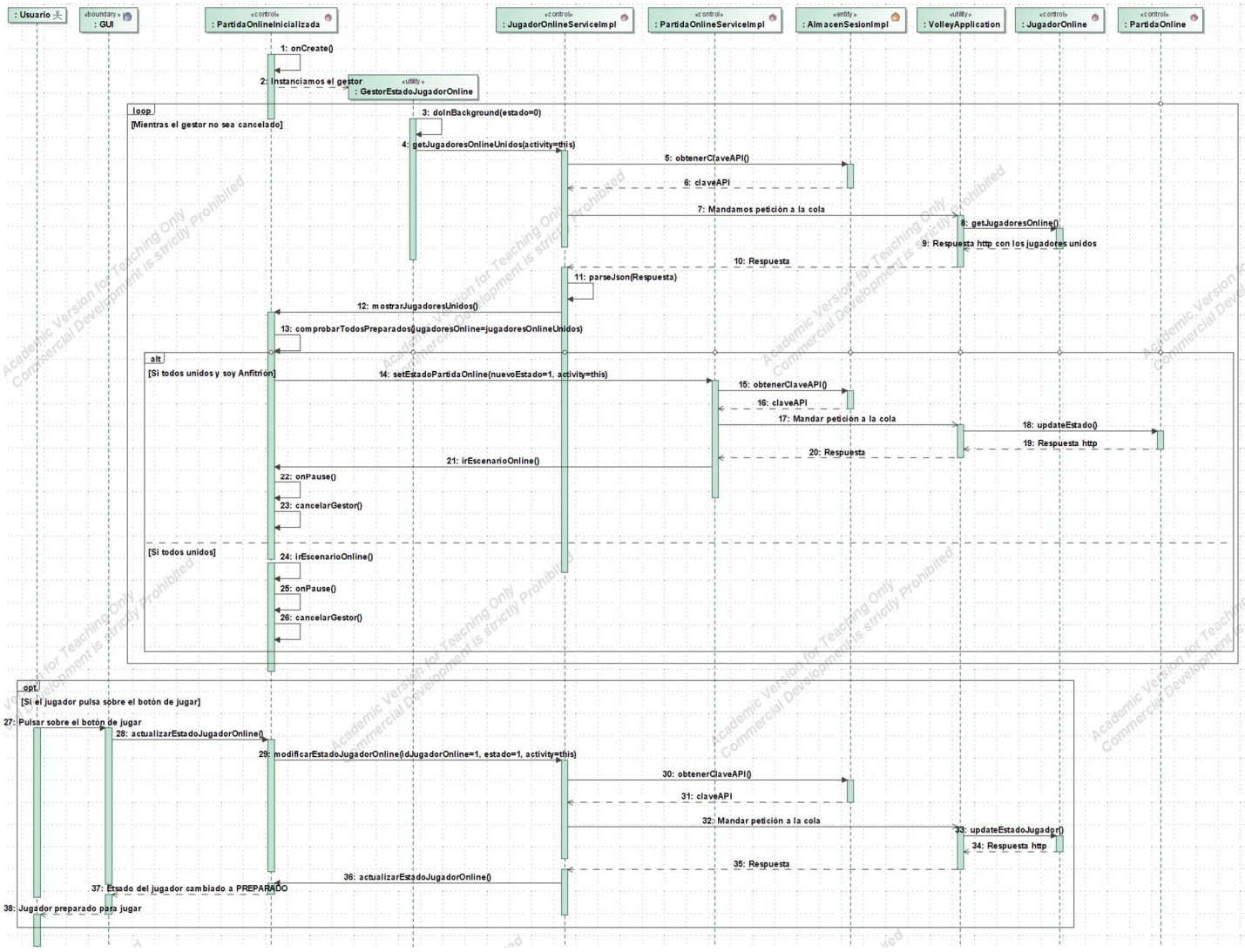


Figura: 41. Diagrama de secuencia de partida online, segunda parte

En esta segunda parte, se muestra los mensajes intercambiados cuando el jugador ya se encuentra dentro de una partida. En esta pantalla podrá ver los jugadores unidos a la partida y si están preparados o no (la información se muestra dinámicamente gracias al uso de los gestores de estado). En caso de que el estado de todos los participantes sea *Preparado*, dará comienzo la partida arrancando la primera ronda (se mostrará la actividad *EscenarioOnline*).

4.1.5.9 Diagramas de estado

Como hemos visto en el diagrama de clases, las clases *JugadorOnline* y *PartidaOnline* tienen una relación fuerte, por lo tanto, si una partida online es eliminada, también serán eliminados aquellos jugadores online referenciados. Una partida online es eliminada cuando el jugador marcado como *Anfitrión* sale de la partida, o cuando ésta ha finalizado.

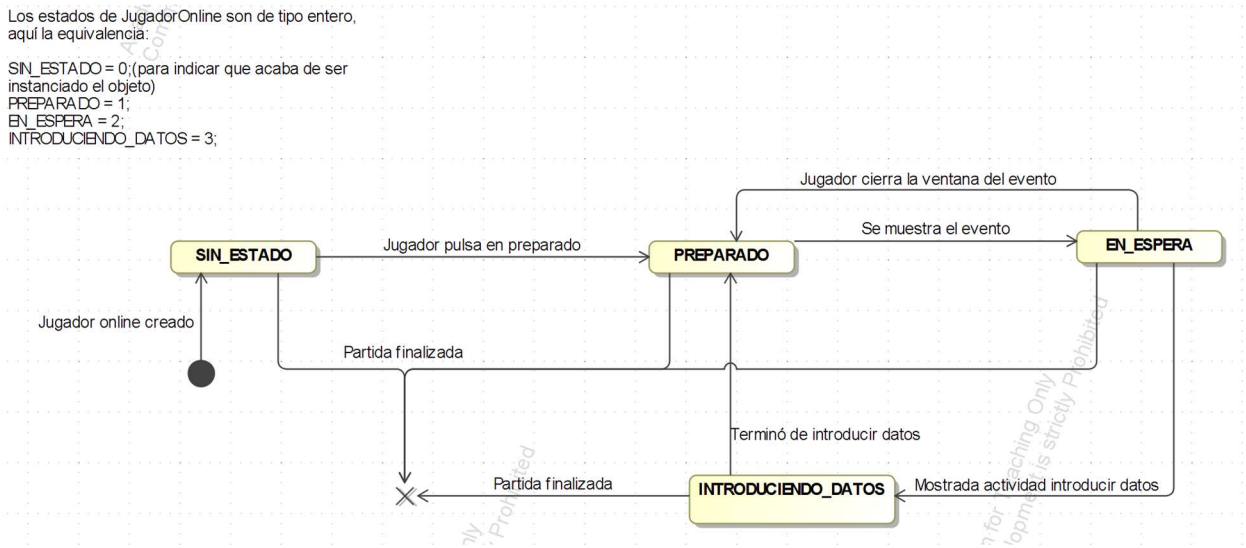


Figura: 42. Diagrama de estados de partida online, JugadorOnline

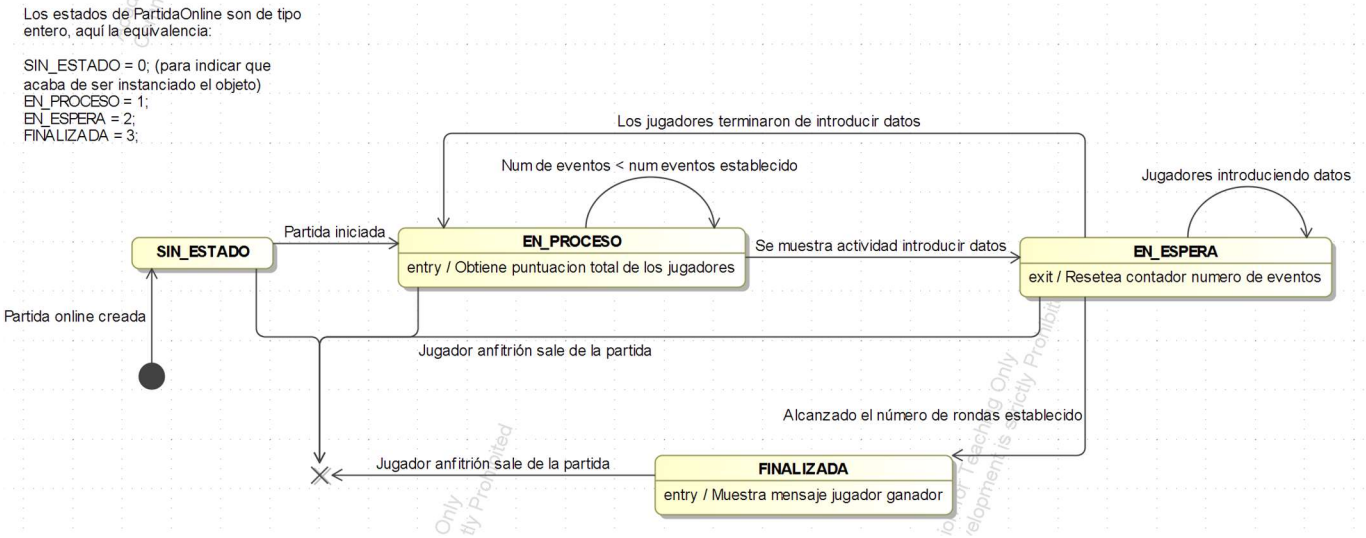


Figura: 43. Diagrama de estados de partida online, PartidaOnline

4.3.3 Gestión de la comunidad

4.1.5.10 Diagramas de clases

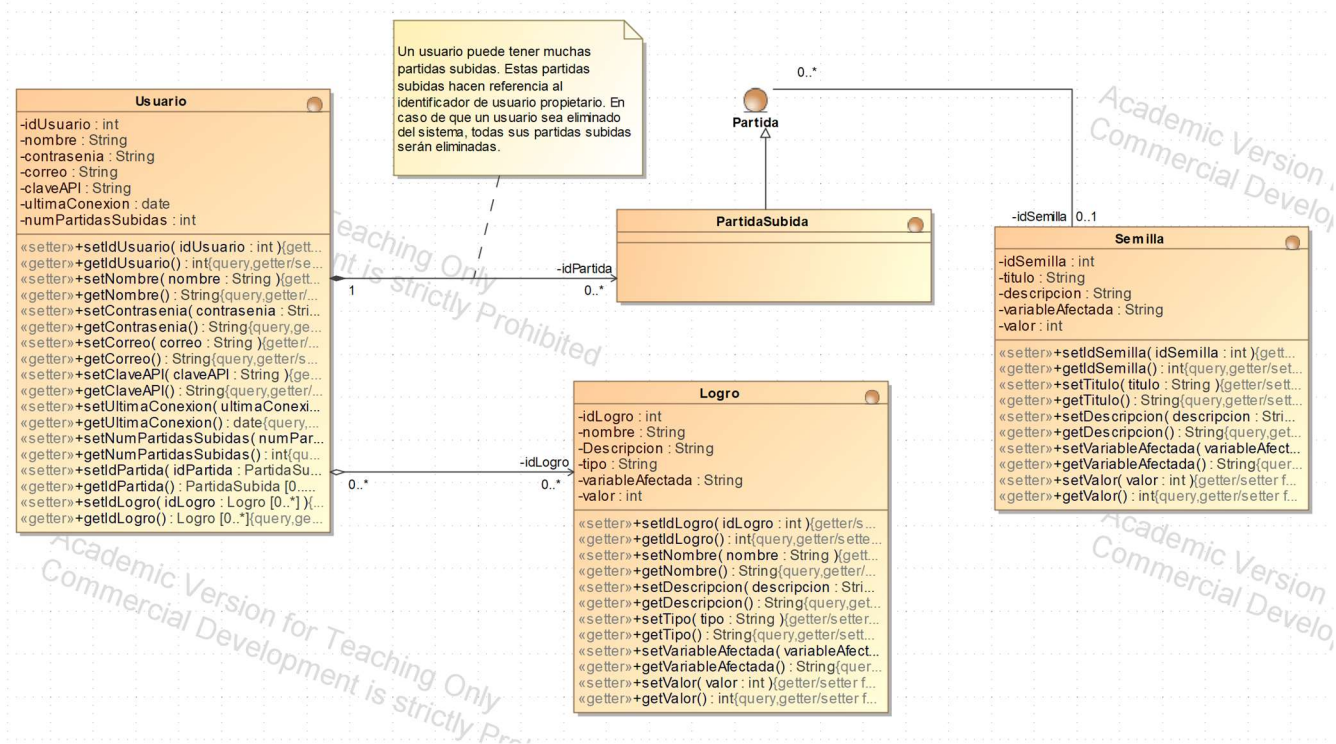


Figura: 44. Diagrama de clases del modelo de comunidad

En este diagrama podemos ver el modelo del subsistema *Gestión de la Comunidad*.

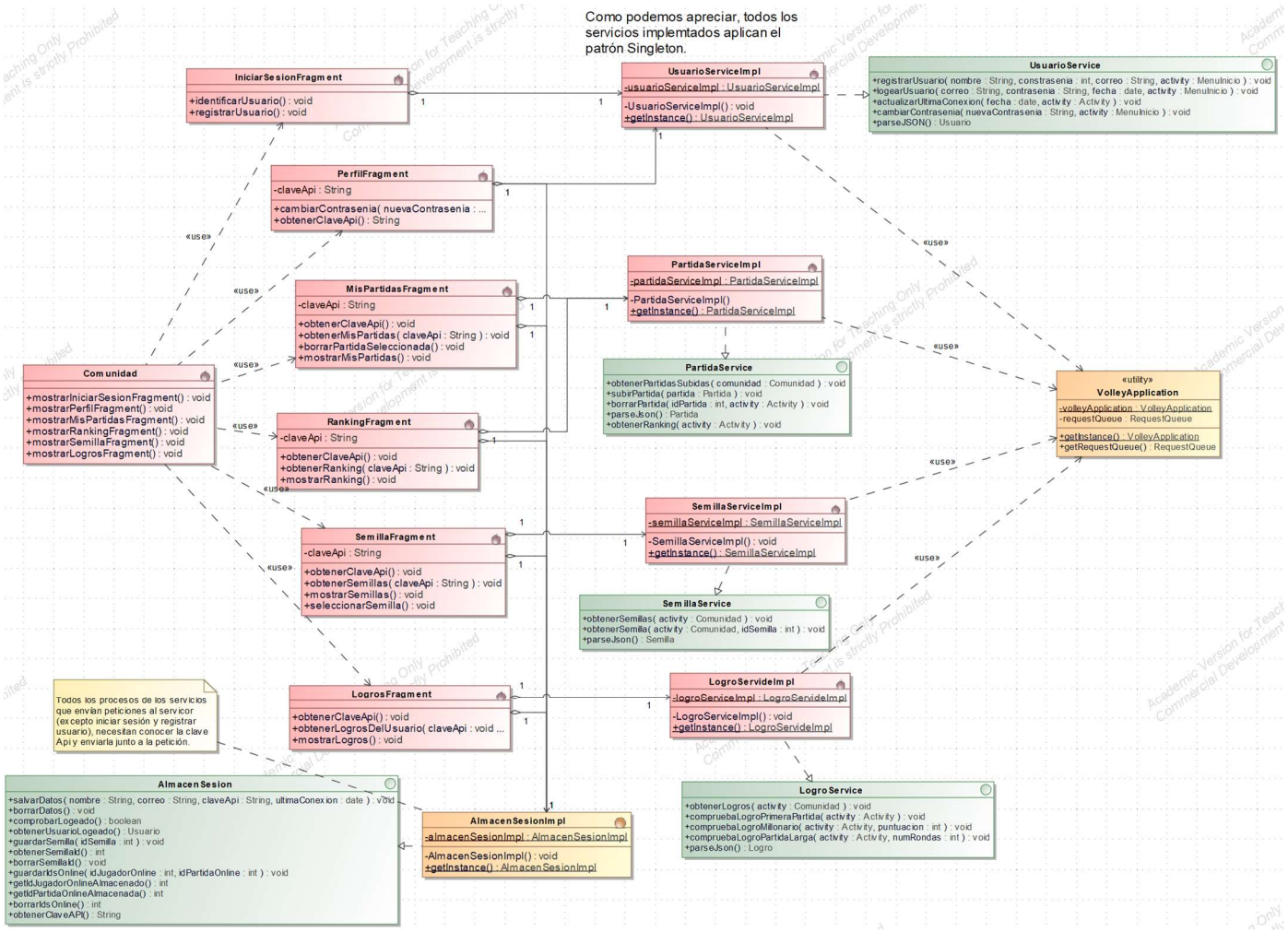


Figura: 45. Diagrama de clases de comunidad

En esta figura podemos concluir que la actividad *Comunidad* es un contenedor que va mostrando el fragmento correspondiente según indique el usuario haciendo uso, por ejemplo, de un menú.

4.1.5.11 Diagramas de objetos

Figura: 46. Diagrama de objetos de comunidad

Esta imagen muestra el momento concreto en el que la BDD remota hay registrado un usuario, quien además subió el 11-11-2018 una partida finalizada. También se muestra el logro *Primera partida compartida* y la semilla con título *Plan de pensiones*.

4.1.5.12 Diagramas de actividad

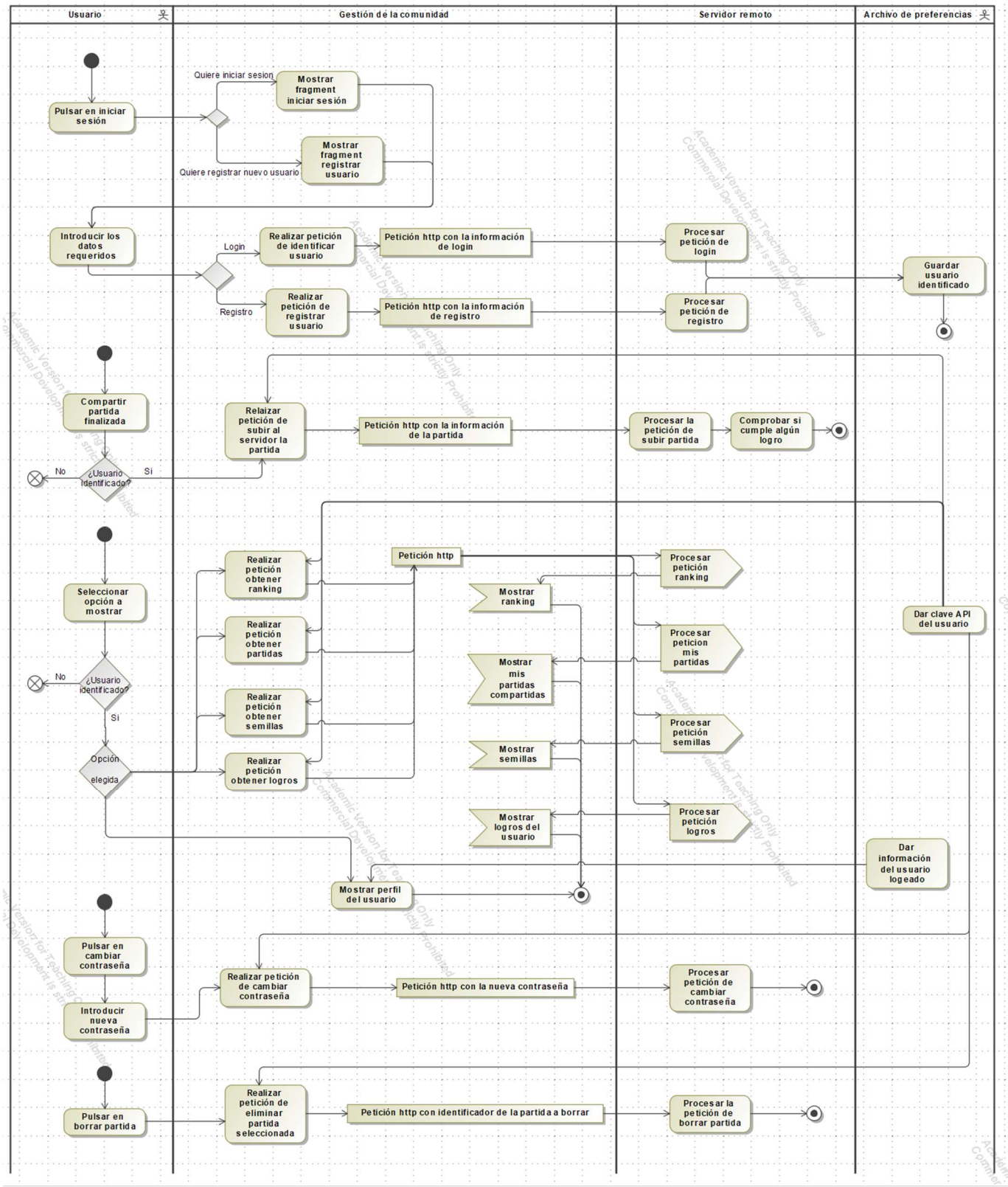


Figura: 47. Diagrama de actividad de comunidad

En este diagrama se representan las distintas acciones (independientes entre sí) que puede realizar el usuario cuando entra en el subsistema comunidad (por medio de su actividad principal). Como podemos ver, cuando realizamos una petición con objetivo de obtener información de la BBDD remota (como pasa en *Mostrar*

ranking), la siguiente actividad de la aplicación no se ejecutará hasta que recibamos las respuestas del servidor y la procesamos, gracias a un método callback.

4.1.5.13 Diagramas de secuencia

Como hemos indicado con anterioridad, todas las respuestas recibidas del servidor serán procesadas gracias al uso de métodos callback localizados en los servicios. Además, éstos pueden activar ciertas partes del comportamiento de una clase controlador.

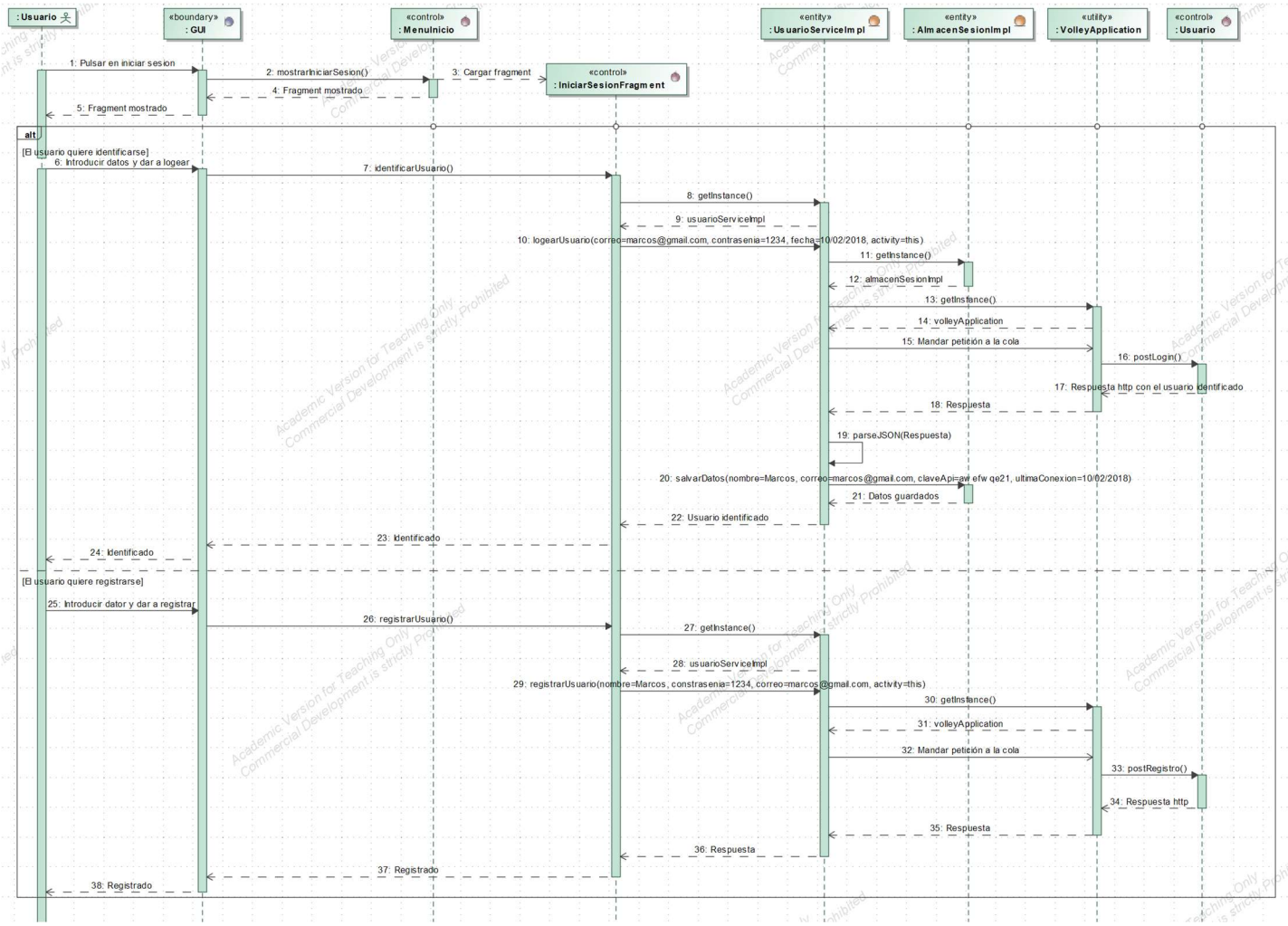


Figura: 48. Diagrama de secuencia de comunidad, inicio de sesión

En este diagrama se representan los principales mensajes intercambiados en el subsistema cuando el *Usuario* quiere iniciar sesión, o registrarse. Recordamos que la llamada a *VolleyApplication* se realiza con un mensaje asíncrono ya que la petición se realiza en un subproceso. Si recibimos la respuesta tras pedir identificarnos, debemos decodificarla para obtener la información de usuario, y a continuación almacenarla de forma persistente.

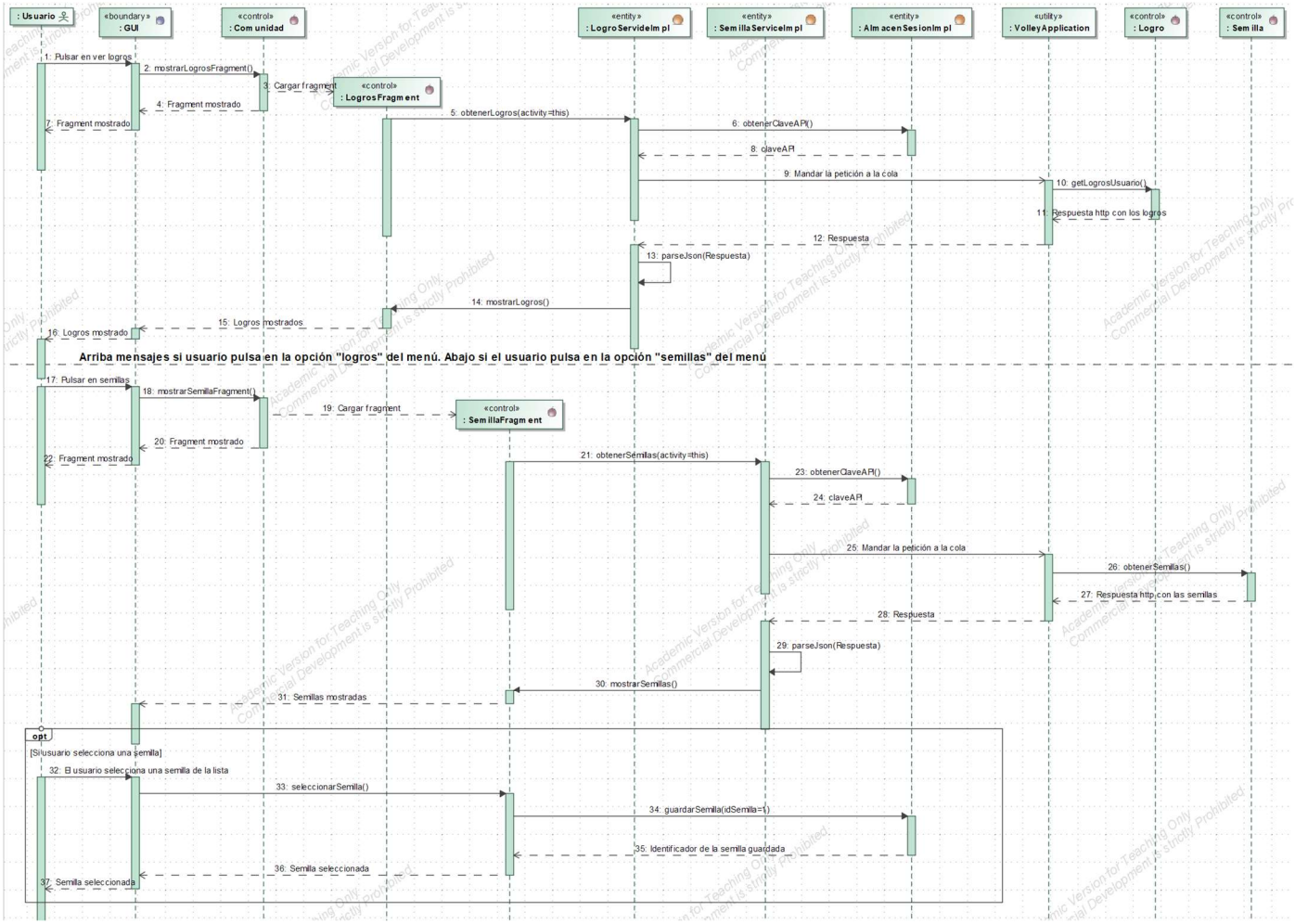


Figura: 49. Diagrama de secuencia de comunidad, logro y semilla

En esta imagen podemos ver los mensajes intercambiados cuando el usuario accede a la sección *logro* y *semilla*. En este caso, cuando obtenemos las semillas disponibles e indicamos que queremos utilizar una en concreto, la aplicación almacenará la información de ésta para ser utilizada más adelante.

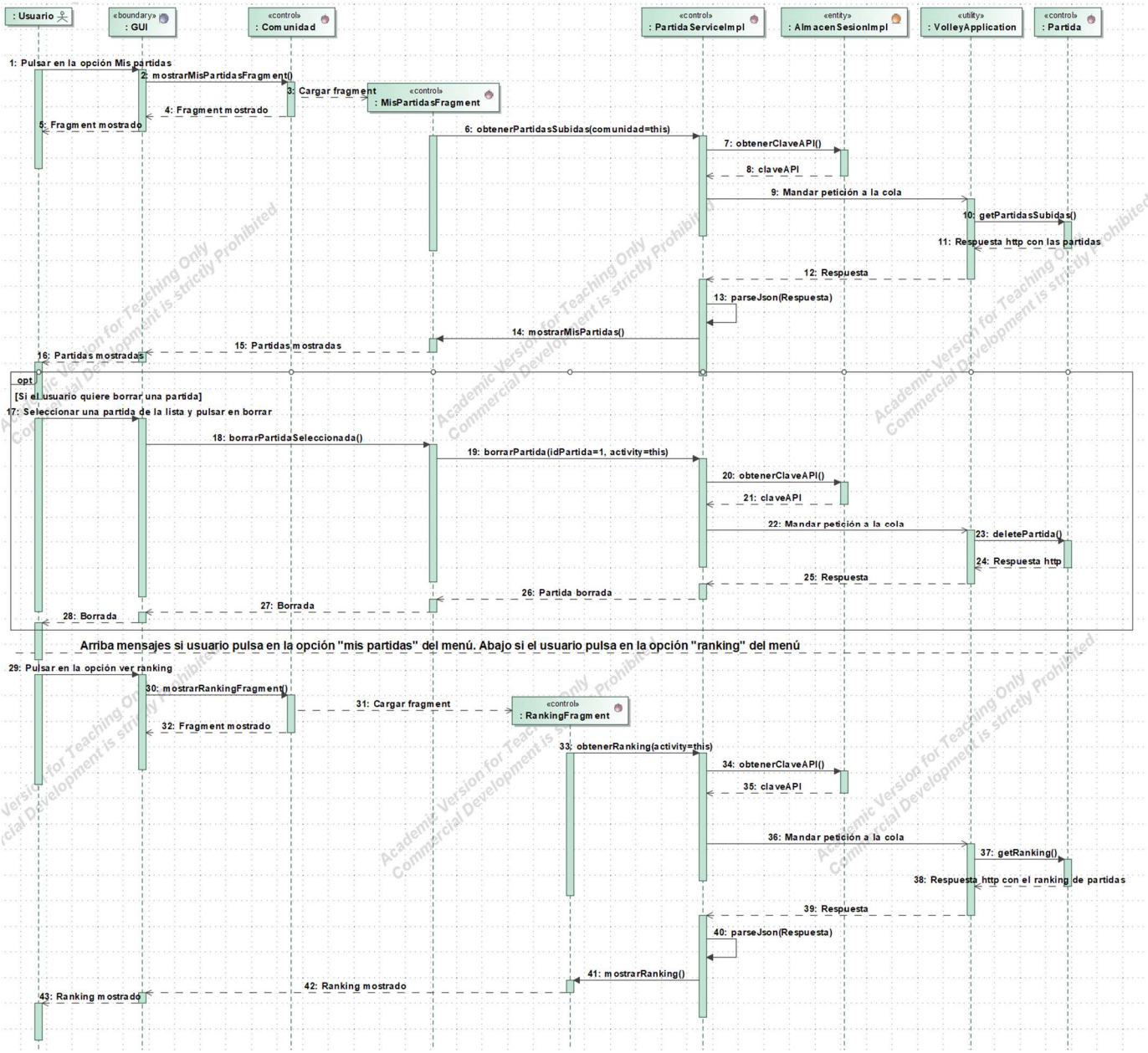


Figura: 50. Diagrama de secuencia de comunidad, ranking y mis partidas

En este caso, en realidad estamos mandando peticiones al servidor sobre el mismo tipo de recurso (*partida*). Sin embargo, accedemos a servicios web distintos que nos devuelven la información solicitada. De forma muy parecida en otras secciones, cuando recibimos los datos, los decodificamos y a continuación llamamos a un método *callback* del controlador para que actualice la información mostrada en pantalla.

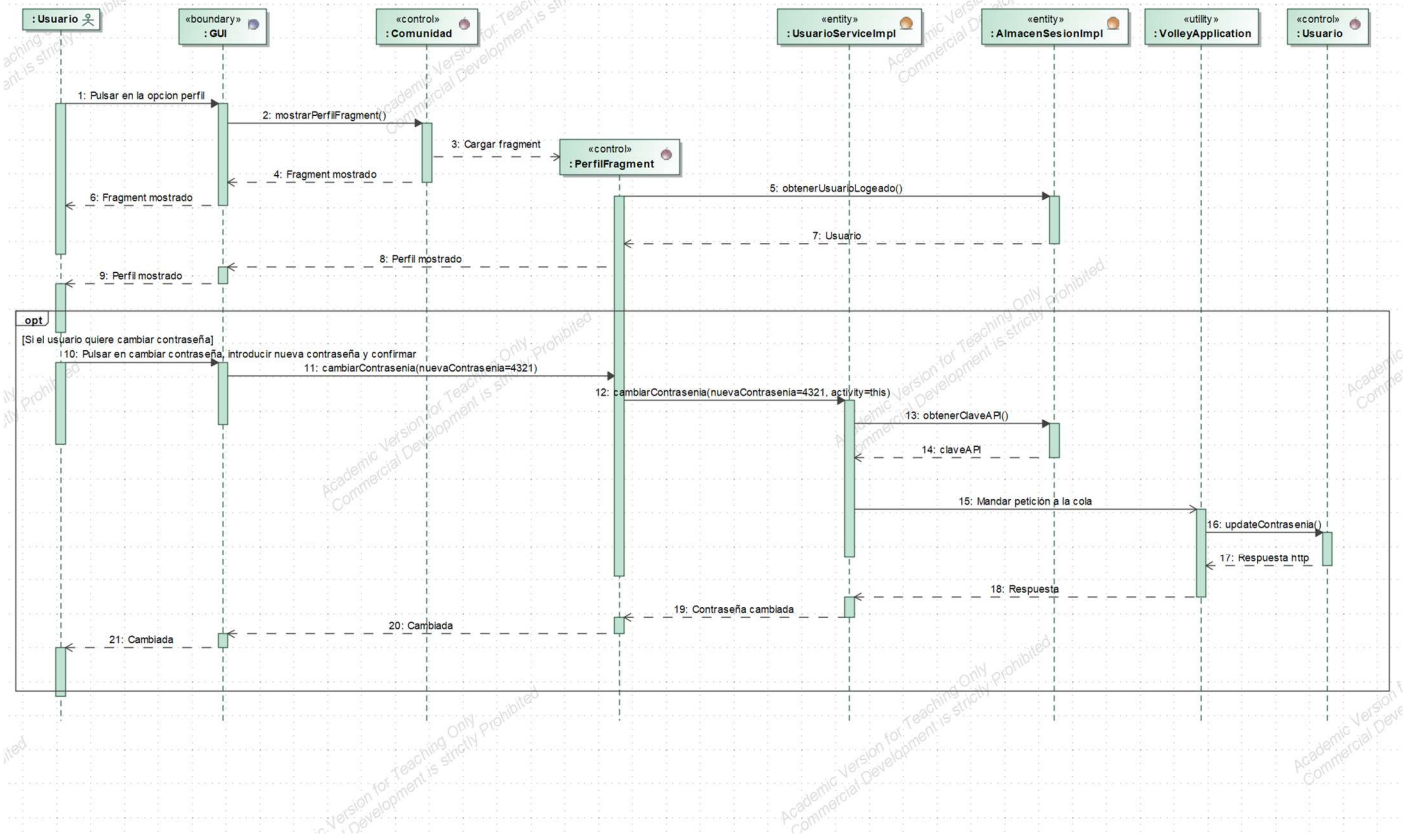


Figura: 51. Diagrama de secuencia de comunidad, perfil

Esta imagen muestra los mensajes intercambiados cuando el usuario accede a la sección *Perfil*. En este caso, la información mostrada no requerirá mandar ninguna petición al servidor ya que los datos a mostrar ya se obtuvieron cuando inició sesión (y fueron almacenados de forma persistente). Sin embargo, si desea cambiar la contraseña, la aplicación sí mandará la petición correspondiente al servidor (con la nueva contraseña).

4.3.4 Consulta de instrucciones y manual de usuario

4.1.5.14 Diagramas de clases

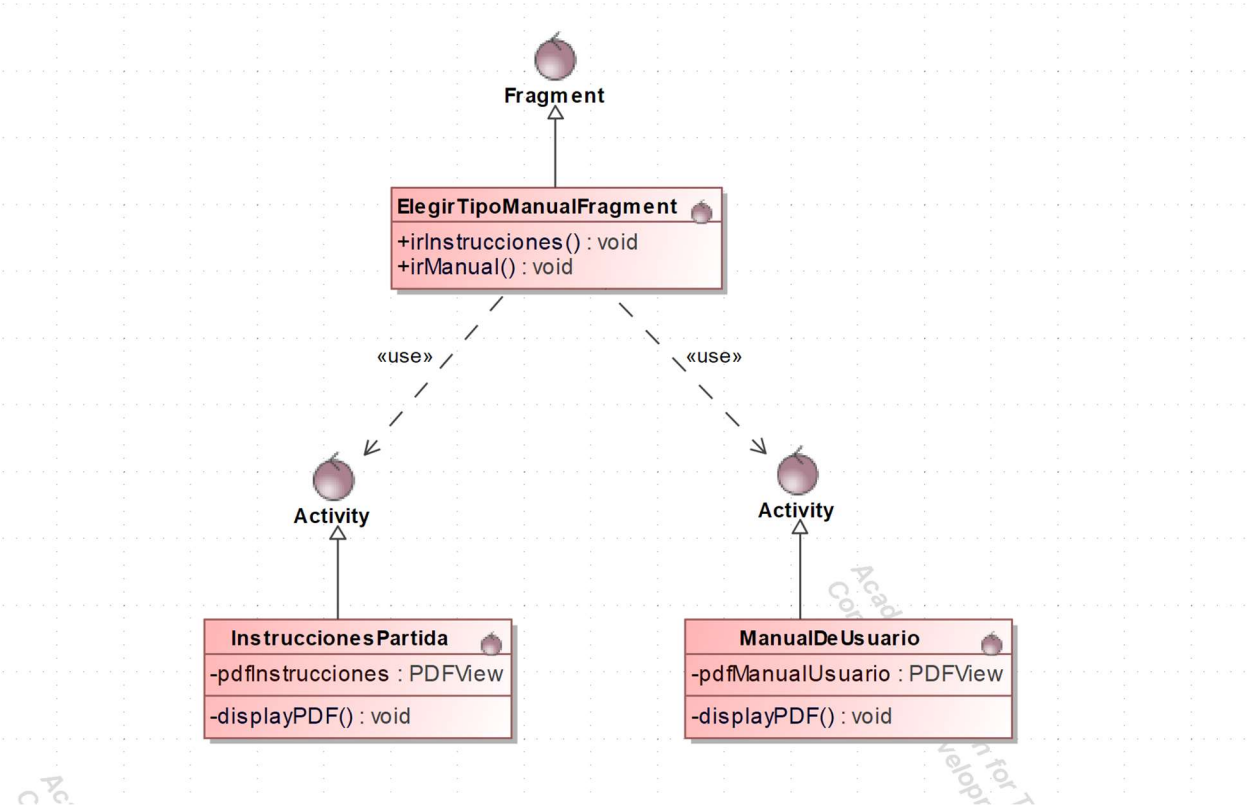


Figura: 52. Diagrama de clase de instrucciones y manual

Este diagrama muestra las clases controlador del subsistema, y sus relaciones.

4.1.5.15 Diagramas de actividad

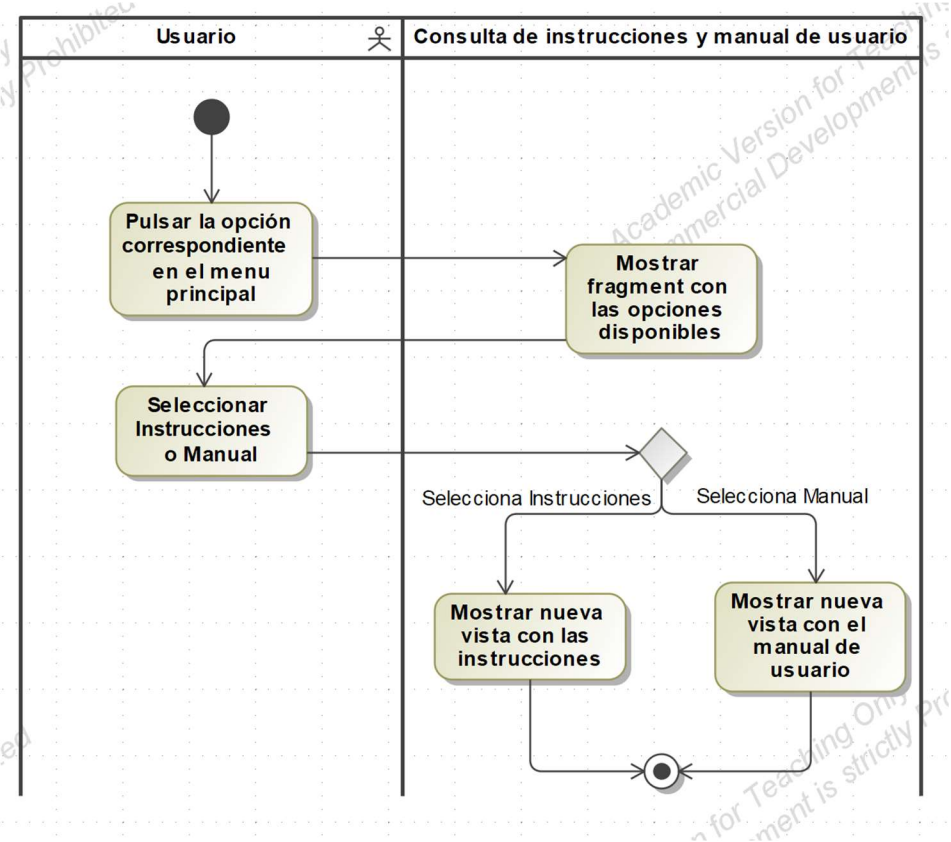


Figura: 53. Diagrama de actividad de instrucciones y manual

En este diagrama podemos ver la secuencia de acciones cuando el usuario quiere visualizar las instrucciones o el manual. Se puede deducir que la aplicación sólo muestra uno de los dos.

4.1.5.16 Diagramas de secuencia

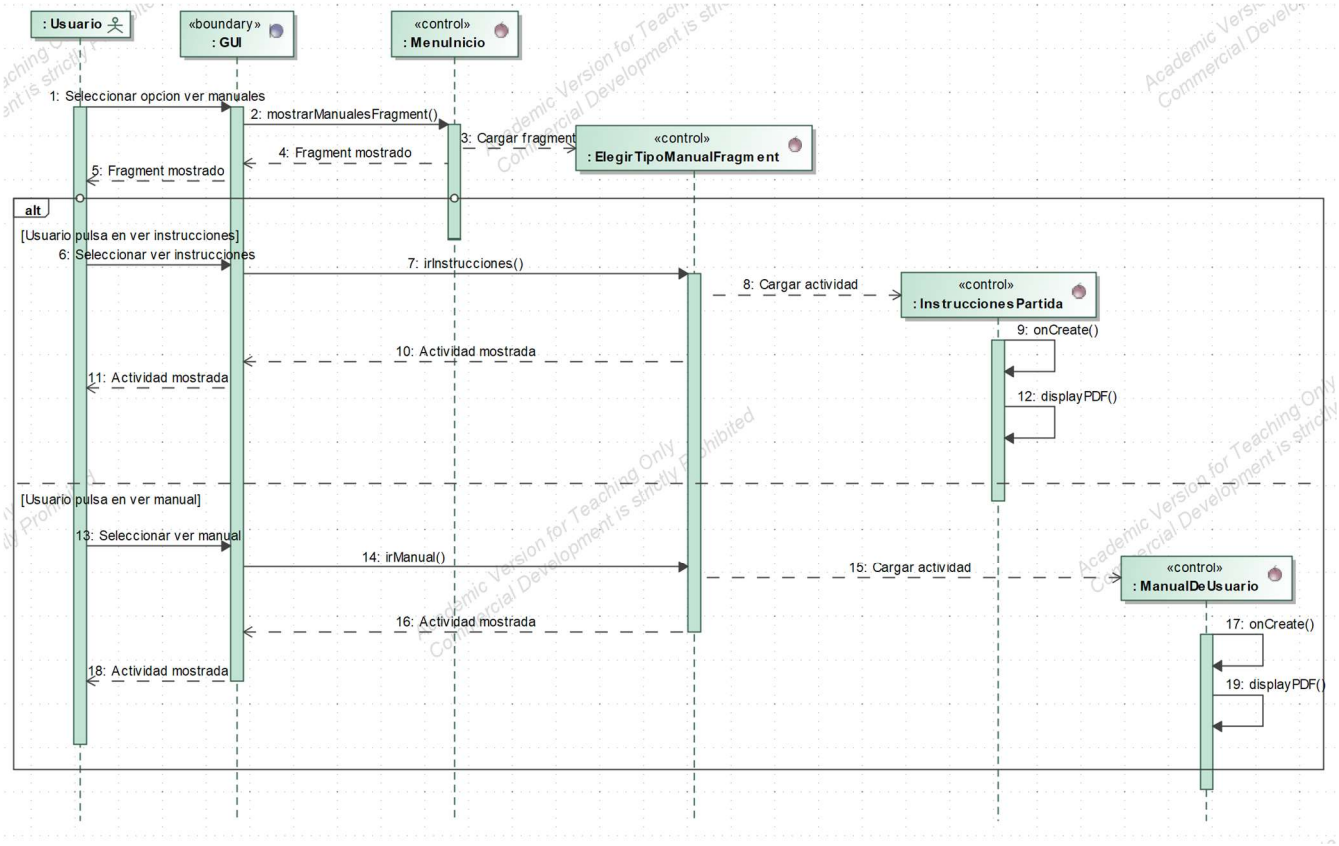


Figura: 54. Diagrama de secuencia de instrucciones y manual

En el siguiente diagrama se muestran los mensajes intercambiados cuando el usuario quiere visualizar las instrucciones o el manual. Cuando selecciona una opción, se lanza la actividad correspondiente. Automáticamente, cuando esta se crea, muestra el documento pdf gracias a *displayPDF()*.

4.3.5 Servidor remoto

4.1.5.17 Diagramas de clase

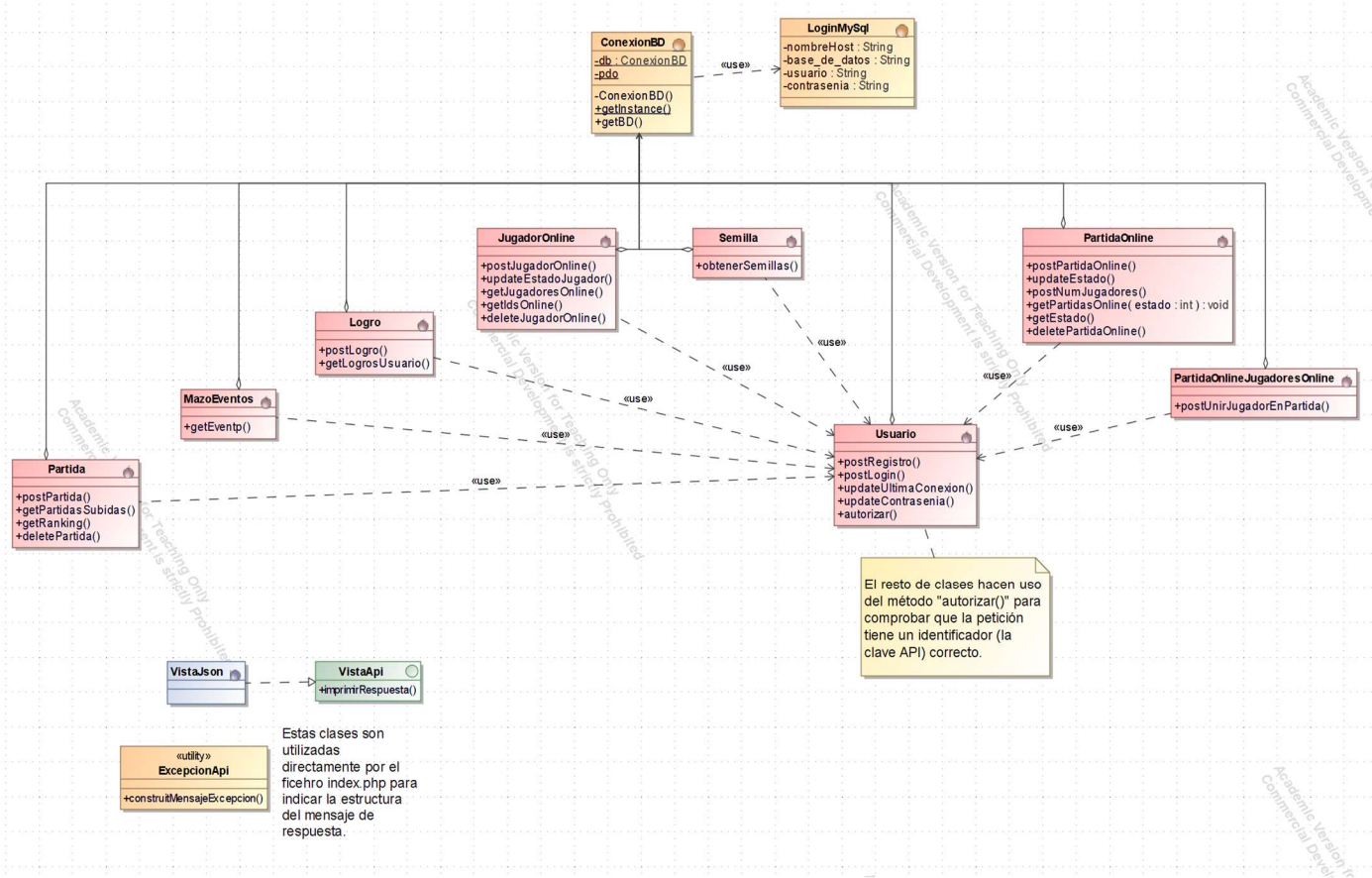


Figura: 55. Diagrama de clases de servidor remoto

En el servidor remoto, los controladores tienen el mismo nombre de las tablas de la BBDD remota que gestionan. Todas las respuestas enviadas por el servidor tienen la misma estructura JSON, la cual está definida en la clase *VistaJson*.

5 IMPLEMENTACIÓN

Continuando con el ciclo de vida del desarrollo software, en este capítulo tratamos la implementación del sistema, que corresponde la programación de la aplicación software y la codificación de los servicios web.

5.1 Arquitectura del Proyecto

En este apartado vamos a tratar cómo se ha organizado los proyectos en Android Studio, y el proyecto correspondiente al desarrollo de los servicios web en PhPStorm. La estructura de ambos proyectos toma como referencia el patrón Modelo Vista Controlador (MVC), agrupando las clases y archivos que realizan funciones del mismo rol, generando proyectos modulares y con componentes reutilizables.

5.1.1 Aplicación Android

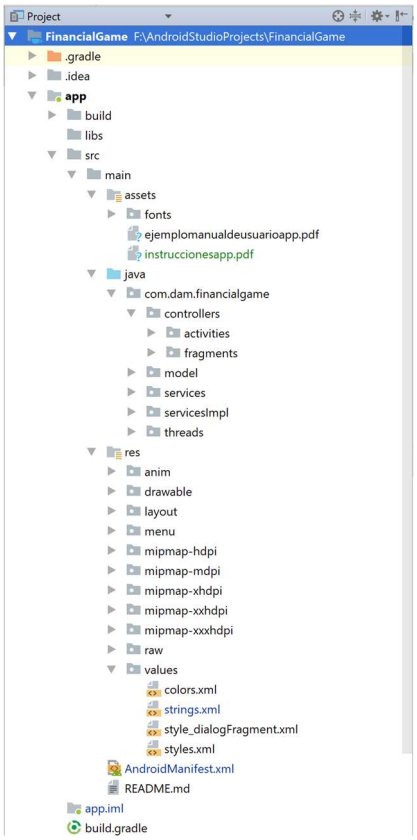


Figura: 56. Arquitectura del proyecto Android

En la imagen mostrada se muestra la arquitectura del proyecto de Android Studio. En la carpeta *java* se encuentran las clases del programa, diferenciadas según su funcionalidad. En la carpeta *controllers* se encuentran las distintas actividades y fragmentos que gestionan los eventos y elementos de la interfaz de usuario (realizan el rol de controlador). En la carpeta *model* se sitúan las clases java que representan los distintos tipos de datos específicos de la aplicación, y en *services* las clases que los gestionan (ambos realizan el rol de modelo). En *threads* podemos encontrar los hilos que gestionan procesos independientes, y la clase *VolleyApplication*. Finalmente, en la carpeta *res/layout*, estarán contenidos los distintos ficheros xml que representan la interfaz de usuario (realizan el rol de vista).

En la carpeta *res/drawable* se almacenan las imágenes e iconos de la aplicación. El fichero

res/values/strings.xml es de gran importancia ya que es un fichero de configuración donde tenemos definidas las url de los servicios web, además de otras cadenas de caracteres que haremos referencia en multitud de componentes del programa. El documento *build.gradle* ya lo hemos nombrado en el apartado de librerías externas, en él indicamos las dependencias de librerías externas, y otras opciones de configuración de la aplicación. Finalmente, dentro de *assets/fonts* guardamos las instrucciones y el manual en formato pdf.

5.1.2 El servidor

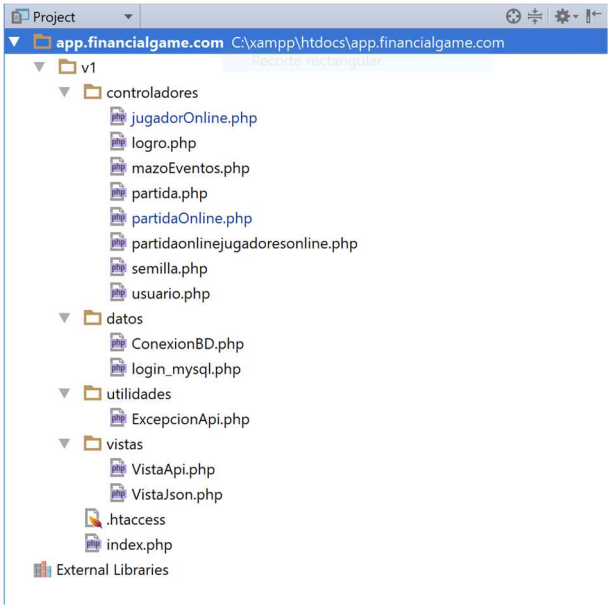


Figura: 57. Arquitectura del proyecto servicios web

En la imagen mostrada se representa la estructura del proyecto php correspondiente al desarrollo de los servicios web. Se muestra claramente el rol de los distintos documentos ya que se han agrupado con este propósito. Debemos indicar que cada clase php de controladores representa una tabla de la base de datos del servidor. El archivo *index.php* redireccionará la petición, según el recurso requerido, a la clase controladora de ese recurso. El objetivo del documento *.htaccess* es indicar que se realice un redireccionamiento a *index.php*.

5.2 La aplicación Financial Game

A continuación, se explica con detalle la aplicación desarrollada, su funcionamiento y opciones que ofrece. Esta información también se mostrará en el documento *Manual de Usuario* ubicado en el programa, el cual los usuarios podrán visualizar libremente.

5.2.1 Acceder a las funciones online

Antes de empezar y explicar los apartados más importantes de la aplicación, debemos resaltar que el programa ofrece un gran número de opciones online como compartir una partida, ver información de perfil, seleccionar un escenario preestablecido, ránking, etc. Para todo ello, es fundamental estar registrado.

5.2.2 Crear una cuenta

Para crear una cuenta en Financial Game, únicamente se necesita un correo válido, contraseña y nombre de usuario. No se olvide del correo introducido y contraseña ya que son las credenciales para luego iniciar sesión.

En la pantalla principal, seleccione en el símbolo de *Usuario* situado en la esquina superior izquierda. Aparecerá el diálogo de iniciar sesión. Pulse en *REGISTRAR* y le aparecerá el siguiente diálogo:

Figura: 58. GUI Crear cuenta

Introduzca los datos requeridos y confirme. Si todo ha ido bien, aparecerá un pequeño mensaje confirmando la acción. Ahora sólo tiene que iniciar sesión y disfrutar de la experiencia que ofrece Financial Game.

5.2.3 Menu principal

Al menú principal se accederá automáticamente después de una pantalla de carga inicial. En esta sección, podrás:

- Acceder a las opciones de iniciar sesión y registro.
- Iniciar una partida offline.
- Iniciar una partida online.
- Entrar en la comunidad online de Financial Game.
- Ver los manuales de la aplicación (instrucciones y manual de usuario).

Recuerde que, para acceder a la comunidad y a las partidas online, deberá estar registrado e identificado.

Figura: 59. GUI Menu principal

5.2.4 Partida offline

5.2.4.1 Iniciar partida

En esta pantalla, deberán introducir los nombres de los participantes y el número de rondas de la partida. Podrá visualizar los jugadores registrados según se introducen. Finalmente, para comenzar la partida, pulse en el botón *Play*.

Figura: 60. GUI Iniciar partida offline

5.2.4.2 Escenario

En esta pantalla la ronda ya ha sido iniciada, podrán ver los jugadores un ránking de las puntuaciones de cada jugador. Para iniciar temporizador pulsar en el botón *Play*.

El temporizador está representado en un diálogo emergente, mostrando una barra de avance según pase el tiempo. Recuerde que es un período aleatorio, por lo tanto, no dudéis mucho en vuestras acciones. Al finalizar el temporizador, se activará un sonido para avisar a todos los jugadores.

Figura: 61. GUI Escenario 1

Al finalizar el temporizador, aparecerá un diálogo con un evento. Después de realizar las acciones requeridas, podrán cerrarlo pulsando en el botón *Cerrar* o bien fuera de la ventana.

Figura: 62. GUI Escenario 2

5.2.4.3 Introducción de datos

En esta sección, los jugadores deberán introducir sus puntuaciones según los activos financieros que tengan y su valor actual. Antes de todo, alguien deberá introducir una serie de valores globales a todos (los mostrados en pantalla). Simplemente escriba los valores indicados. Cuando termine, pulse en la siguiente opción del menú superior.

Figura: 63. GUI Introducir datos 1

Aquí se mostrará los distintos activos financieros para que cada jugador indique el número que tiene. Deben asegurarse de que el nombre indicado corresponda con el jugador que está introduciendo los datos. En las opciones *Efectivo* y *Financiación* sí deberán introducir manualmente la cantidad exacta. Cuando haya terminado, pulsar en *Aceptar*. Saldrá un diálogo de confirmación, pulse en *Confirmar* si está seguro, o *Cancelar* si desea comprobar los datos.

Figura: 64. GUI Introducir datos 2

5.2.4.4 Fin de la partida y resumen

Figura: 66. GUI Fin

Cuando se hayan introducido los datos de todos los jugadores, y se hayan alcanzado el número de rondas indicado, aparecerá el diálogo de fin del juego. Pulse en *Ver gráficas* para pasar a la pantalla final de partida offline.

En esta sección, según la opción elegida en el menú superior, los jugadores podrán ver información valiosa del transcurso de la partida representada en gráficas.

La primera muestra el avance de las puntuaciones totales de cada jugador en cada ronda, podrá diferenciar cada participante gracias a los colores de la gráfica. La segunda muestra cómo se compone la puntuación final de cada jugador, representado cada producto financiero en un color concreto. Podrá alternar entre jugadores pulsando en sus nombres.

Figura: 65. GUI Graficas

Finalmente, podrá compartir la partida en la comunidad de Financial Game, para ello simplemente pulse *Subir Partida*. Recuerde que, para acceder a esta funcionalidad, **deberá estar identificado previamente** antes de iniciar la partida offline.

5.2.5 Partida online

5.2.5.1 Crear partida online o unirse a una

En esta pantalla podrá unirse a una partida ya creada, o crear una usted mismo.

Para unirse a una partida ya creada, selecciónela de la lista y automáticamente será redirigido a la pantalla de esa partida.

En caso de que desee crearla, pulse en *Crear Partida* y seleccione los valores deseados de las opciones mostradas. Pulse en *Crear* y será redirigido automáticamente a la pantalla de esa partida.

Figura: 67. GUI Partida online

5.2.5.2 Partida online inicializada

En esta pantalla se mostrará la pantalla principal de la partida online seleccionada/creada. Podrá ver un listado de los jugadores unidos, quién es el anfitrión (quien ha creado la partida), el número de jugadores máximos permitidos, y el número de rondas de la partida.

Si usted ya está preparado para empezar a jugar, pulse en el botón *Play*. Verá que, en el listado, se actualizará su estado a PREPARADO. Cuando todos los jugadores estén preparados, dará comienzo la partida.

Figura: 68. GUI Partida online inicializada

5.2.5.3 Escenario de la partida online

Cuando haya empezado la partida, será redirigido a esta pantalla. Aquí, como en la partida offline, podrá llevar el ritmo de la partida.

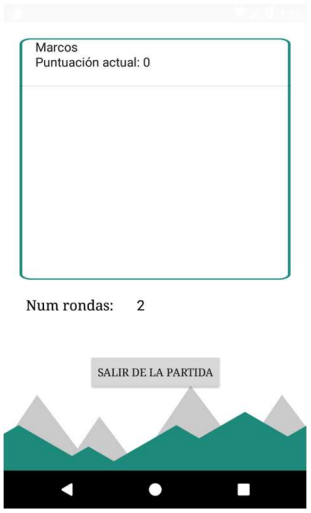


Figura: 69. GUI Escenario online

5.2.6 Comunidad

En esta sección, podrá alternar entre las distintas opciones pulsando en el menú superior izquierdo de la pantalla.

Figura: 70. GUI Comunidad

5.2.6.1 Perfil

Opción por defecto al entrar en esta sección. Aquí podrá ver información básica de su perfil, además de cambiar la contraseña.

Para este último caso, pulse en *Cambiar Contraseña*, y luego *OK* para confirmar. Si la modificación ha sido correcta, aparecerá un mensaje emergente confirmando la acción.

Figura: 71. GUI Perfil

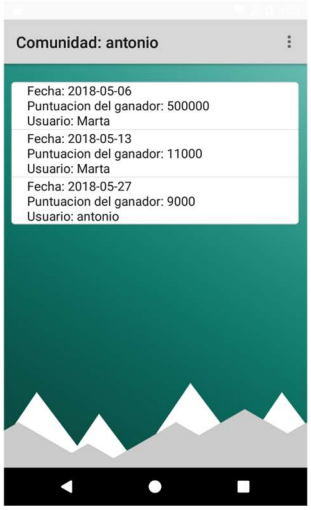
5.2.6.2 Mis partidas

En esta pantalla se muestran las partidas compartidas por usted. Si pulsa sobre una de ellas podrá ver más información.

Además, se le ofrecerá borrarla. Para ello, pulse en el botón *Borrar*. Se actualizará automáticamente la lista.

Figura: 72. GUI Partidas

5.2.6.3 Ranking



En esta pantalla se muestra un listado con las primeras 20 partidas ordenadas de mayor a menor puntuación, indicando la fecha y el usuario que las compartió.

Figura: 73. GUI Ranking

5.2.6.4 Selección de escenario preestablecido

En esta pantalla, se le muestra los distintos escenarios preestablecidos que usted puede utilizar para sus partidas offline. Para ver más información de cada uno, selecciónelo de la lista.

Si quiere utilizarlo, seleccione la opción *Utilizar semilla*. Si quiere cancelarlo, simplemente vuelva a seleccionar la opción.

Recuerde que debe leer con atención la descripción y efecto, ya que indica a los jugadores si deben cambiar algún valor de inicialización de la partida, o si no es necesario ya que modifica valores internos de la aplicación.

Figura: 74. GUI Escenario preestablecido

5.2.6.5 Logros

En esta pantalla, se le muestra los logros obtenidos por usted al compartir una partida offline finalizada. Hay dos listas.

En la primera se clasifica los logros obtenidos según el tipo, representado en medallas de bronce, plata y oro. Un logro de tipo Oro es más complicado de conseguir que uno de tipo Plata, y a su vez el tipo Bronce es más fácil que los demás.

En la segunda simplemente se ordenan los logros obtenidos.

Figura: 75. GUI Logros

5.2.7 Manuales

A continuación, se le muestra en pantalla el diálogo emergente para elegir qué tipo de manual quiere visualizar.

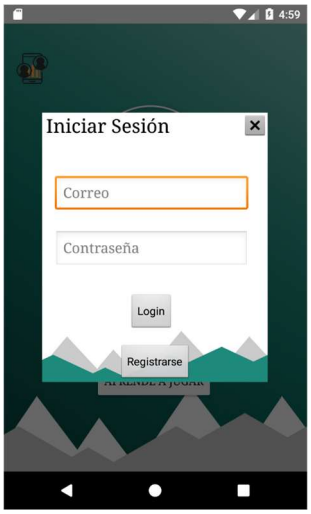
Figura: 76. GUI Manuales

En las siguientes imágenes aparecen las instrucciones y el manual de usuario. Cuando esté viendo los documentos, podrá hacer zoom y ver la página mostrada.

Figura: 77. GUI Manual

Figura: 78. GUI Instrucciones

5.2.8 Inicio de sesion



Para poder iniciar sesión será necesario haberse registrado antes (apartado 1.1 de este manual).

En la pantalla principal de la aplicación, pulse sobre el icono *Usuario*. A continuación, se mostrará la ventana de inicio de sesión. Introduzca los datos y pulse *Login*.

Se cerrará automáticamente y podrá ver que ha cambiado el símbolo de la esquina superior izquierda. Ahora se muestra el incono *Salir de la sesión* y el nombre de usuario. Para cerrar sesión pulse sobre el símbolo comentado.

Si usted se identifica, y cierra la aplicación, al volver a iniciarla seguirá estando identificado.

Figura: 79. GUI Iniciar sesion

6 PLAN DE PRUEBAS

En este capítulo se explica la estrategia de pruebas seguida para la validación de los requisitos. Las pruebas serán realizadas por el autor del proyecto fin de carrera, y se llevarán a cabo paralelamente al desarrollo del software. Las herramientas utilizadas para realizar las pruebas han sido el entorno de programación AndroidStudio para probar la aplicación, Advanced REST client (una aplicación de Google Chrome) para probar los servicios web, y XAMPP para crear un servidor de prueba pero que cumpla con los requisitos. Debemos recordar que los datos enviados en las peticiones, o los que recibimos en la respuesta del servidor, están en formato JSON.

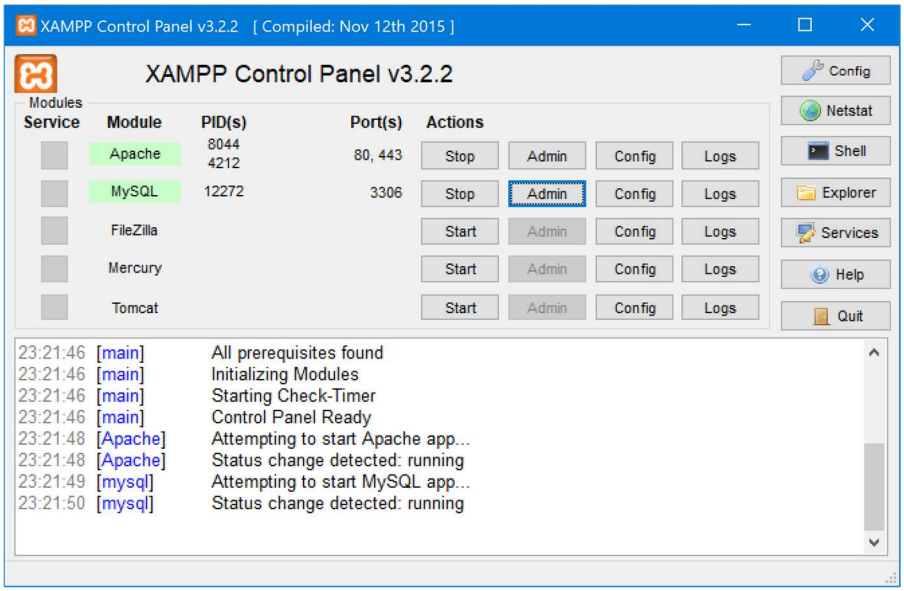


Figura: 80. Captura del programa XAMPP

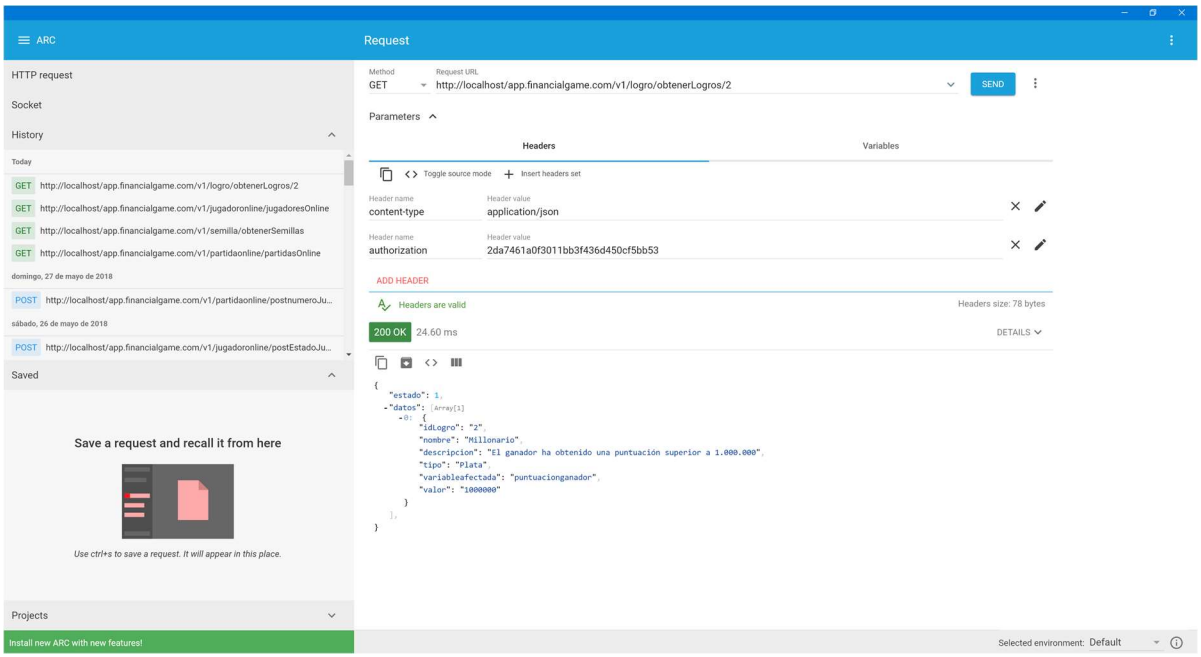


Figura: 81. Captura del programa Advanced REST client

El plan de pruebas se organizará según los subsistemas definidos. El encargado de realizar las pruebas hará el papel del actor *Usuario*, y modificará los valores de *número de rondas* y el *intervalo del temporizador* para agilizar el proceso. Finalmente, se propondrá pruebas de aceptación para que el cliente pueda comprobar si el software cumple las expectativas.

6.1 Gestión de partida offline

CP-01	Inicializar partida
Descripción:	Se probará que los participantes puedan introducir sus nombres, elegir número de rondas y que se crea una partida con esa información.
Casos de uso asociados:	CU04
Prerrequisitos:	Disponible la actividad *InicializarPartidaOffline*.
Resultado esperado:	Los datos introducidos se almacenan en la BBDD local.

CP-02	Temporizador
Descripción:	Se probará que la actividad *Escenario* lance el temporizador en un hilo aparte, obtenga el valor aleatorio comprendido entre el intervalo establecido, y se actualice la barra de avance mostrada en pantalla según avanza el tiempo.
Casos de uso asociados:	CU02
Prerrequisitos:	Disponible la actividad *Escenario*, y la clase *TemporizadorThread* implementada.
Resultado esperado:	Se activa el temporizador y se actualizará correctamente la barra de estado.

CP-03	Evento
Descripción:	Se probará que se muestre un evento en la actividad *Escenario*. Para ello, el servicio *PartidaOfflineService* deberá coger de la BBDD local un objeto de la clase evento, y devolvérselo al controlador.
Casos de uso asociados:	CU02
Prerrequisitos:	Disponible la actividad *Escenario*, el servicio *PartidaOfflineService* implementado y también la clase *Evento*.

Resultado esperado:	Se muestra en pantalla el evento elegido, con sus atributos asociados.

CP-04	**Guardado de las puntuaciones introducidas**
Descripción:	Se probará que los datos introducidos en la actividad *IntroducirDatos* se guarden en la BBDD local, y se asocien correctamente al jugador deseado.
Casos de uso asociados:	CU03
Prerrequisitos:	Disponible la actividad *IntroducirDatos*, y el servicio *PartidaOfflineService* implementado.
Resultado esperado:	Las puntuaciones de cada jugador se almacenan correctamente en la BBDD local, y corresponden con los escritos.

CP-05	**Cambios debido a la semilla elegida**
Descripción:	Se probará que, cuando un jugador introduce sus datos, si hay alguna variable afectada (un producto financiero) por la semilla distribuida ésta cambia al valor especificado.
Casos de uso asociados:	CU06
Prerrequisitos:	Disponible la actividad *IntroducirDatos*, implementado el servicio *PartidaOfflineService*.
Resultado esperado:	Los valores de los productos financieros afectados por la semilla se modifican, y se aplican en el cómputo de la puntuación total del jugador.

CP-06	**Visualización de las gráficas de la partida**
Descripción:	Se probará que se representen correctamente las gráficas que muestran información de la partida finalizada. Deben de poder verse las rondas jugadas, las puntuaciones totales de cada jugador, y el valor de cada producto financiero que la componen.
Casos de uso asociados:	CU05
Prerrequisitos:	Disponible la actividad *GraficasJugadores*, implementado el servicio *PartidaOfflineService*, disponibles los fragmentos *LineGraphFragment* y

	PieGraphFragment.
Resultado esperado:	Se mostrará las gráficas con la información correcta, y permitirá poder navegar entre sus opciones.

CP-07	**Partida compartida**
Descripción:	Se probará que un usuario puede subir al servidor la partida offline jugada.
Casos de uso asociados:	CU07
Prerrequisitos:	Actividad *GraficasJugadores* disponible, servicio *PartidaOfflineService* implementado, servicio *PartidaService* implementado, el servicio web ofrecido por el servidor probado y funcionando.
Resultado esperado:	Se subirá al servidor la partida deseada, creándose un nuevo registro en la tabla *Partida* de la BBDD remota.

6.2 Gestión de partida online

CP-08	**Incorporación de jugador a la partida**
Descripción:	Se probará que el usuario se incorpore a la partida online seleccionada.
Casos de uso asociados:	CU09
Prerrequisitos:	Actividad *PrincipalPartidaOnline* disponible, servicio *PartidaOnlineService* implementado, las clases *PartidaOnline* y *JugadorOnline* desarrolladas, el servicio web ofrecido por el servidor probado y funcionando.
Resultado esperado:	El usuario se unirá a la partida esperada, pasando a la siguiente actividad *PartidaOnlineInicializada*, y creándose su respectivo jugador en la tabla *JugadorOnline*.

CP-09	**Crear partida online**
Descripción:	Se probará que un usuario pueda crear una partida online, introduciendo una serie de parámetros como el número de rondas y el número de jugadores máximos permitidos.

Casos de uso asociados:	CU10
Prerrequisitos:	Actividad *PrincipalPartidaOnline* disponible, fragment *CrearPartidaOnlineFragment* disponible, servicio *PartidaOnlineService* implementado, las clases *PartidaOnline* y *JugadorOnline* desarrolladas, el servicio web ofrecido por el servidor probado y funcionando.
Resultado esperado:	Se registrará en la BBDD remota la nueva partida online, con los parámetros deseados.

CP-10	**Recarga dinámica de las partidas disponibles**
Descripción:	Se comprobará que la lista de partidas online disponibles, mostrada en la actividad *PrincipalPartidaOnline*, se recargue dinámicamente, actualizándose la información periódicamente.
Casos de uso asociados:	CU09
Prerrequisitos:	Actividad *PrincipalPartidaOnline* disponible, servicio *PartidaOnlineService* implementado, la clase *PartidaOnline* desarrollada, el gestor *GestorEstadoPartidaOnline* desarrollado, el servicio web ofrecido por el servidor probado y funcionando.
Resultado esperado:	Periódicamente, el gestor envía una petición preguntando por las partidas online creadas, y con esa información se actualizará el listado.

CP-11	**Inicio de la partida**
Descripción:	Se comprobará que no se empieza la partida online hasta que todos los jugadores estén preparados.
Casos de uso asociados:	CU10
Prerrequisitos:	La actividad *PartidaOnlineInicializada* disponible, el servicio *JugadorOnlineServide* programado, la clase *JugadorOnline* desarrollada, el gestor *GestorEstadoJugadorOnline* implementado, el servicio web ofrecido por el servidor probado y funcionando.
Resultado esperado:	No se mostrará la actividad *Escenario* online en el dispoitivo de los jugadores hasta que no estén todos preparados.

6.3 Gestión de la comunidad

CP-12	Iniciar sesión
Descripción:	Se probará que, desde la aplicación, e introduciendo los datos requeridos, pueda iniciar sesión una persona. En caso de introducir datos erróneos, deberá mostrarse un mensaje emergente en la pantalla. Además, en caso positivo, deberá almacenarse en local la información de usuario.
Casos de uso asociados:	CU19
Prerrequisitos:	La actividad *MenuPrincipal* disponible, el fragment *IniciarSesionFragment* disponible, los servicios *UsuarioService* y *AlmacenSession* desarrollados, la clase *Usuario* programada, el servicio web ofrecido por el servidor probado y funcionando.
Resultado esperado:	Una persona podrá identificarse en caso de introducir datos correctos, se capturará la respuesta del servidor, y se almacenará la información del usuario en el archivo de preferencias.

CP-13	Registrar usuario
Descripción:	Se probará que una persona pueda registrar un nuevo usuario, escribiendo el correo, nombre y contraseña. Para ello, el correo introducido deberá estar en un formato correcto (loquesea@loquesea.loquesea).
Casos de uso asociados:	CU20
Prerrequisitos:	La actividad *MenuPrincipal* disponible, el fragment *IniciarSesionFragment* disponible, el servicio web ofrecido por el servidor probado y funcionando.
Resultado esperado:	En caso de introducir un correo con formato no válida, aparecerá un mensaje de error. En caso positivo, aparecerá mensaje confirmando la acción y en la BBDD remota aparecerá el nuevo registro en la tabla *Usuario*.

CP-14	Ver perfil del usuario
Descripción:	Se comprobará que al acceder a la actividad *Comunidad*, por defecto aparezca la información de perfil del usuario identificado.

Casos de uso asociados:	CU21
Prerrequisitos:	La actividad *Comunidad* disponible, el fragmento *PerfilFragment* disponible, el sevicio *AlmacenSession* implementado, la clase *Usuario* desarrollada.
Resultado esperado:	Se muestra la información de perfil del usuario.

CP-15	**Ver ranking**
Descripción:	Se probará que al entrar en la sección *Ranking* de la actividad *Comunidad*, se muestre el ranking obtenido de la respuesta de la petición correspondiente al servidor.
Casos de uso asociados:	CU22
Prerrequisitos:	La actividad *Comunidad* disponible, el fragment *RankingFragment* disponible, el servicio *PartidaService* implementado, la clase *Partida* desarrollada, el servicio web ofrecido por el servidor probado y funcionando.
Resultado esperado:	Se capturará la respuesta del servidor, cogemos el listado de partidas que compone el ranking, y se mostrará en la pantalla.

CP-16	**Semilla distribuida**
Descripción:	Se probará que la entrar en la sección *Escenario preestablecido* de *Comunidad* se muestra un listado con todas las semillas disponibles. Además, al pulsar en una, se almacena en local la información de ésta.
Casos de uso asociados:	CU08
Prerrequisitos:	La actividad *Comunidad* disponible, el fragment *SemillaFragment* disponible, los servicios *SemillaService* y *AlmacenSession* implementados, la clase *Semilla* desarrollada, el servicio web ofrecido por el servidor probado y funcionando.
Resultado esperado:	Se representarán en la pantalla el listado de semillas disponible, además al pulsar sobre una, se guardará su información en el archivo de preferencias.

CP-17	Logros
Descripción:	Se probará que la entrar en la sección *Logros* de *Comunidad* se muestra un listado con todos los logros disponibles, agrupados según el tipo.
Casos de uso asociados:	CU23
Prerrequisitos:	La actividad *Comunidad* disponible, el fragment *LogroFragment* disponible, el servicio *LogroService* implementado, la clase *Logro* desarrollada, el servicio web ofrecido por el servidor probado y funcionando.
Resultado esperado:	El usuario podrá ver sus logros obtenidos, para ello capturamos la respuesta del servidor y cogemos el listado de logros devueltos.

6.4 Consulta de instrucciones y manual de usuario

CP-18	Visualizar manuales
Descripción:	Se probará que una persona pueda acceder a las actividades *Instrucciones* y *ManualDeUsuario* desde el menú principal de la aplicación.
Casos de uso asociados:	CU36, CU37
Prerrequisitos:	Las actividades *MenuPrinicipal*, *Intrucciones* y *ManualDeUsuario* disponibles, documentos pdf de prueba almacenados en la carpeta *resources* del proyecto Android.
Resultado esperado:	Se podrá seleccionar una de las dos opciones desde el menú principal y se redirigirá automáticamente a la actividad, se mostrará el documento elegido, navegar por él y hacer zoom.

6.5 Servidor de la aplicación

CP-19	Usuario registrado
Descripción:	Se probará que, con la herramienta Advanced REST client, el servicio web correspondiente a registrar usuario funcione correctamente, registrando en la BBDD remota el nuevo usuario introducido. Para ello debemos indicar la URL del recurso y poner los valores en el cuerpo del

	mensaje http en formato JSON (correo, nombre, contraseña).
Casos de uso asociados:	CU29, CU30, CU31
Prerrequisitos:	Servicio web del registro desarrollado y activo en el servidor.
Resultado esperado:	Se creará un nuevo registro en la tabla *Usuario* con los datos mandados en la petición.

CP-20	**Identificar usuario**
Descripción:	Se probará que, con la herramienta Advanced REST client, el servicio web correspondiente a identificar usuario funcione correctamente, recibiendo del servidor la respuesta con la información del usuario. Para ello debemos indicar la URL del recurso y poner los valores en el cuerpo del mensaje http en formato JSON (correo, contraseña).
Casos de uso asociados:	CU32, CU33
Prerrequisitos:	Servicio web del usuario desarrollado y activo en el servidor.
Resultado esperado:	En caso de introducir un correo o contraseña erróneos, recibir mensaje de error. Si ponemos correo y contraseña existentes, recibiremos una respuesta confirmando la acción y con la información del usuario.

CP-21	**Servicio Logro**
Descripción:	Se probará, con la herramienta Advanced REST client, el servicio web correspondiente a obtener los logros del usuario. Para ello debemos indicar la URL del recurso e indicar en la cabecera del mensaje http el parámetro *authorization* con valor una claveAPI.
Casos de uso asociados:	CU28
Prerrequisitos:	Servicio web de logro desarrollado y activo en el servidor.
Resultado esperado:	En caso de enviar una claveAPI que no pertenece a ningún usuario del sistema, el servidor responderá con un error. En caso contrario, recibiremos el listado de logros de ese usuario en formato JSON.

CP-22	Servicio Semilla
Descripción:	Se probará, con la herramienta Advanced REST client, el servicio web correspondiente a obtener las semillas registradas en la BBDD remota. Para ello debemos indicar la URL del recurso e indicar en la cabecera del mensaje http el parámetro *authorization* con valor una claveAPI.
Casos de uso asociados:	CU26
Prerrequisitos:	Servicio web de la semilla desarrollado y activo en el servidor.
Resultado esperado:	En caso de enviar una claveAPI que no pertenece a ningún usuario del sistema, el servidor responderá con un error. En caso contrario, recibiremos el listado de semillas en formato JSON.

CP-23	Servicio partida
Descripción:	Se probará, con la herramienta Advanced REST client, los servicios web correspondientes a subir partida, obtener partida y borrar partida. Para ello debemos indicar la URL del recurso e indicar en la cabecera del mensaje http el parámetro *authorization* con valor una claveAPI. En caso de subir una partida debemos indicar en el cuerpo del mensaje http los datos de ésta, y si queremos borrarla el identificador.
Casos de uso asociados:	CU27
Prerrequisitos:	Servicios web de la partida desarrollado y activo en el servidor.
Resultado esperado:	En caso de enviar una claveAPI que no pertenece a ningún usuario del sistema, el servidor responderá con un error. Si queremos borrar una partida e indicamos un identificador no válido, también recibiremos un error. Si no, deberemos obtener una respuesta positiva.

CP-24	Servicio partida online
Descripción:	Se probará, con la herramienta Advanced REST client, el servicio web correspondiente a obtener las partidas online con el estado indicado. Para ello, en el cuerpo del mensaje http debemos indicar el estado buscado.

Casos de uso asociados:	CU16
Prerrequisitos:	Servicio web de partida online desarrollado y activo.
Resultado esperado:	En caso de enviar una claveAPI que no pertenece a ningún usuario del sistema, el servidor responderá con un error. En caso contrario, recibiremos las partidas online registradas en la BBDD con identificador igual al indicado.

CP-25	**Servicio jugador online**
Descripción:	Se probará, con la herramienta Advanced REST client, el servicio web correspondiente a crear un jugador online, y el servicio web correspondiente a obtener los jugadores online unidos a la misma partida online que el usuario. Para crear el jugador, debemos indicar en el cuerpo del mensaje http el identificador de la partida online a la que queremos unirnos.
Casos de uso asociados:	CU15
Prerrequisitos:	Servicio web de jugador online desarrollado y activo.
Resultado esperado:	En caso de enviar una claveAPI que no pertenece a ningún usuario del sistema, el servidor responderá con un error. Si indicamos el identificador de una partida online que no existe, también recibiremos un error. Si no, se creará correctamente el jugador online (relacionándolo con la partida indicada), y recibiremos un listado en formato JSON con los jugadores si mandamos petición a ese servicio web.

CP-26	**Comprobación de logros**
Descripción:	Se probará que, cuando subimos una partida al servidor, se le pasará por una serie de filtros para comprobar si cumple con los requisitos de algún logro. En caso positivo, debe crearse ese logro y asociarlo al usuario que compartió la partida.
Casos de uso asociados:	CU35
Prerrequisitos:	Servicio web de jugador online desarrollado y activo, servicio web de subir partida probado y funcionando.

Resultado esperado:	En caso de enviar una claveAPI que no pertenece a ningún usuario del sistema, el servidor responderá con un error. Si no, y la partida cumple con algún logro (por ejemplo, la puntuación ganadora mayor que un valor predeterminado), se le asociará ese logro al usuario.

CP-27	**Autorización del cliente**
Descripción:	Se probará el sistema de autorización para acceder a los servicios web. Para ello, es necesario que, cuando un servicio requiere que el usuario esté identificado, éste envía en la petición su claveAPI asociada.
Casos de uso asociados:	CU34
Prerrequisitos:	El proceso de autorización deberá estar desarrollado, y al menos un servicio web que requiera la identificación del usuario.
Resultado esperado:	Si el usuario no envía una clave API, o la que envía es incorrecta, no tendrá acceso al servicio web. Si no, podrá acceder al servicio web ya que el proceso de autorización le reconocerá.

6.6 Pruebas de aceptación

CP-28	**Alfa**
Descripción:	Se ejecutará una versión de prueba de la aplicación completa, con objetivo de comprobar la disposición de los elementos en la interfaz de usuario.
Casos de uso asociados:	-
Prerrequisitos:	Tener compilada la aplicación, y el apk generado. Disponer de un dispositivo Android o de un emulador.
Resultado esperado:	Se espera que haya una tasa de fallos elevada y el cliente indique el incumplimiento de ciertos requisitos, además de fallos en la interfaz de usuario.

CP-29	**Beta**

Descripción:	Se entregará a los usuarios una interfaz de usuario completa y completamente codificada. Se espera que el cliente pruebe la aplicación en un escenario real (jugando a una partida con varias personas), y periódicamente problemas que hayan encontrado en el uso del programa a la vez que sugerencias o carencias en el uso de las funciones.
Casos de uso asociados:	-
Prerrequisitos:	Tener compilada la aplicación, y el apk generado. Disponer de un dispositivo Android o de un emulador. Conexión a internet. Alojar los servicios web en un hosting gratuito temporal.
Resultado esperado:	Se espera que el usuario puede realizar una partida offline de principio a fin, poder visualizar correctamente los manuales, registrarse en el sistema, iniciar sesión y acceder a las funciones de la comunidad. Si el sistema tuviese fallos graves, se plantearía la entrega de una versión beta mejorada. Habrá tipos de mensajes http, como los *Update* y *Delete* que no serán procesados por el servidor, debido a las limitaciones de su naturaleza gratuita.

7 PUBLICACIÓN DE LA APLICACIÓN

En este apartado se hará una breve descripción del proceso por el cual una aplicación Android se pone a disposición de los usuarios haciendo uso del mercado de apps Play Store. Este proceso consta de dos tareas principales, preparación de la aplicación y el lanzamiento. El proceso de publicación se realiza normalmente cuando se termina de probar la aplicación en un entorno de depuración.

7.1 Preparación de la aplicación

Durante este paso, se compila la versión de lanzamiento de la aplicación, que será la que los usuarios instalarán en sus dispositivos Android. Debe cumplir una serie de exigencias para garantizar la mayor limpieza y eficiencia del programa.

Debemos eliminar todos aquellos elementos de nuestro código destinados a la depuración. También quitaremos los recursos redundantes (imágenes, por ejemplo), y actualizaremos las direcciones URL. Firmaremos la versión de lanzamiento con una llave privada, necesaria para más tarde subir actualizaciones del programa (hay que tener especial cuidado en no perder el almacén de llaves y en olvidar la contraseña), y la compilaremos, generando el apk de la aplicación. Instalaremos este archivo en un móvil y Tablet Android para probar el correcto funcionamiento. Finalmente, nos aseguraremos de que nuestros servicios web son seguros y están listos para la producción.

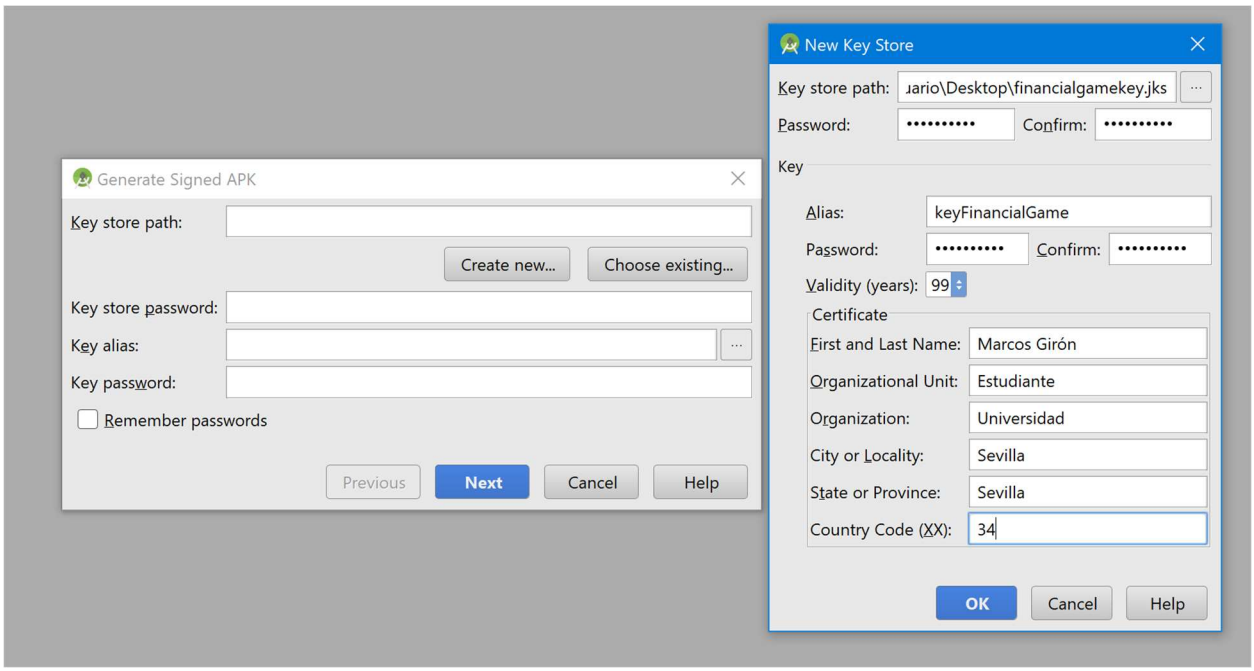


Figura: 82. Proceso de firmar la versión de lanzamiento de Financial Game App

7.2 Lanzamiento

Con la versión de lanzamiento compilada y firmada, procedemos a subir la aplicación a Play Store. Para ello, debemos tener registrada una cuenta de desarrollador en la plataforma Google Play Developer Console (se debe de realizar un pago inicial de 25$).

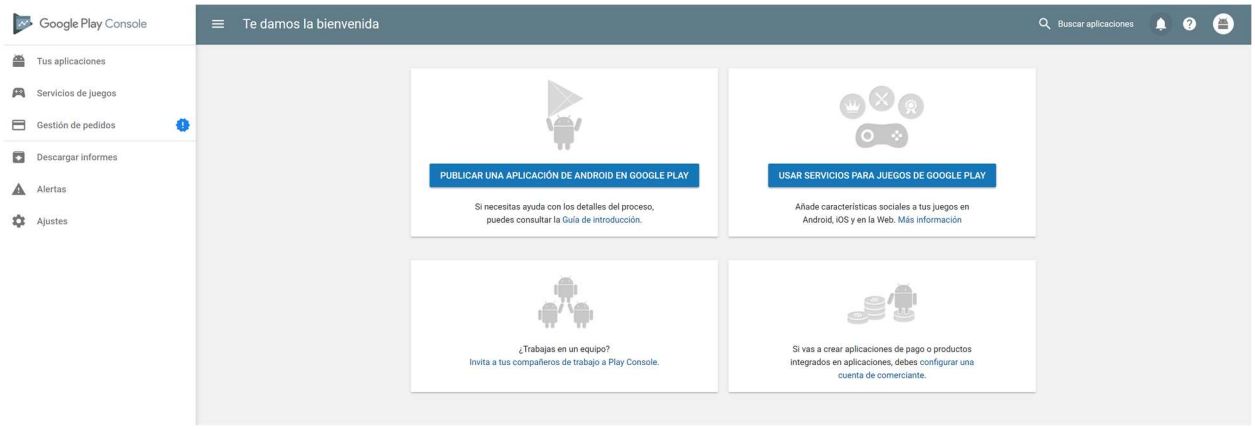


Figura: 83. Página de inicio de Google Play Console

A continuación, pulsamos sobre la opción de publicar la aplicación. Nos pedirá completar una serie de formularios correspondientes a la ficha de Play Store del programa. También se pedirá rellenar la clasificación de contenido de la aplicación.

Después de completar los campos requeridos (no es necesario completarlo todo ya que ofrece la opción de guardar un borrador), podremos acceder al panel de control de la aplicación. En la sección de versiones podremos gestionar en qué estado se va a publicar la aplicación, si en producción (totalmente abierto al público) o en un entorno de prueba abierto donde cualquiera puede participar propietario (éste puede gestionar el número máximo de testers). En esta última opción los participantes podrán reportar fallos al propietario.

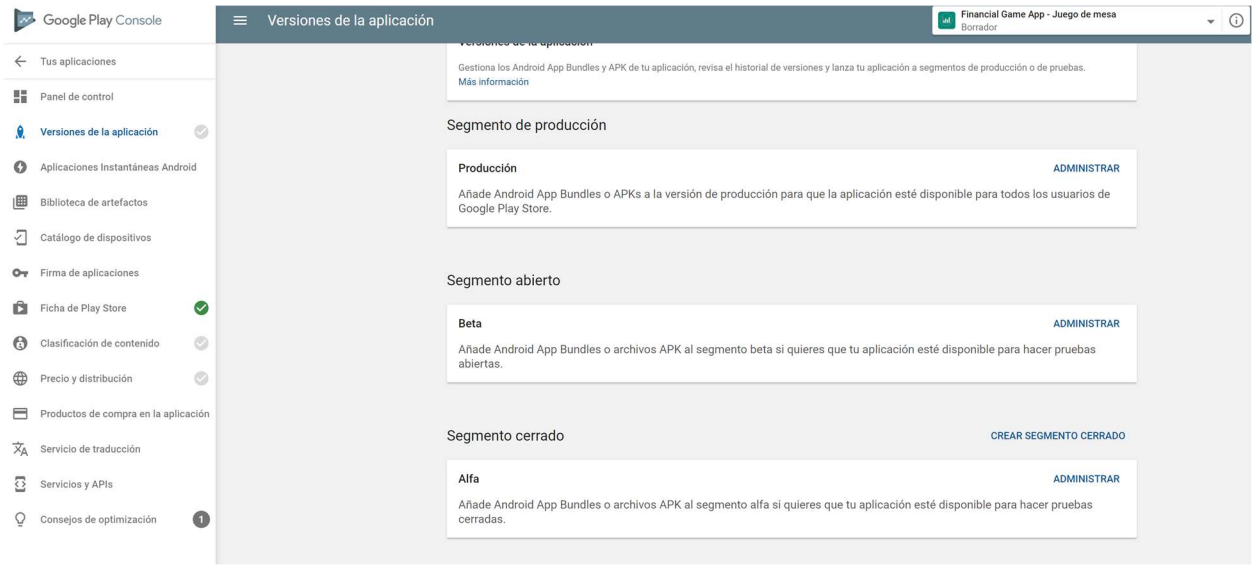


Figura: 84. Panel de control de la aplicación

Debido al carácter pedagógico del proyecto, y las numerosas mejoras que quedan por implementar, se ha decidido subir la aplicación en versión BETA para que, aquellos que decidan acceder al programa, puedan probarlo y reportar errores.

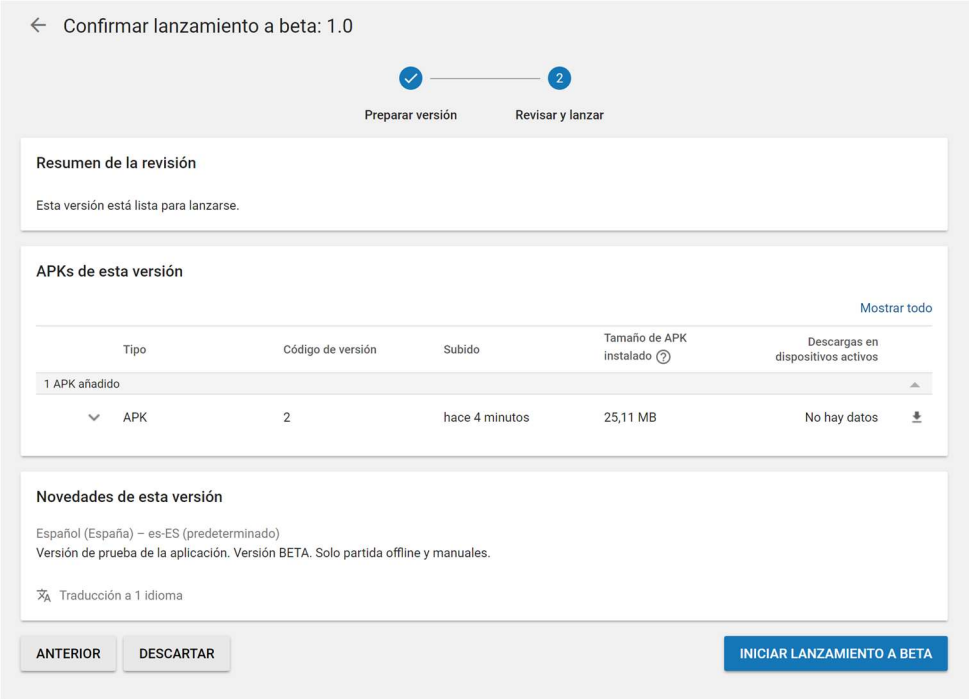


Figura: 85. Lanzamiento de la versión BETA de la aplicación

Finalmente, pulsando la opción de iniciar lanzamiento, tendremos disponible la versión de prueba abierta en la Play Store. Para encontrar las aplicaciones lanzadas con este tipo de versión dentro de la tienda digital, debemos buscar el enlace *acceso beta* en la categoría, y ahí encontrar el programa deseado.

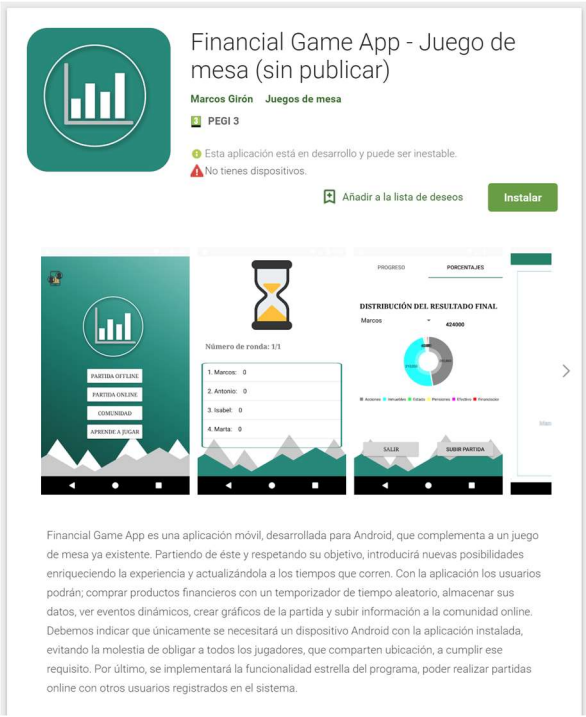


Figura: 86. Financial Game App versión BETA en Play Store

8 CONCLUSIONES

En este último capítulo, trataremos las principales dificultades encontradas durante el transcurso del proyecto y cómo estas han sido resueltas. Seguiremos indicando los posibles pasos a seguir para ampliar, y mejorar, las funciones de la aplicación y el sistema. Finalmente, terminaremos con una conclusión personal.

8.1 Dificultades encontradas

A continuación, se listan los principales problemas encontrados en el desarrollo de este trabajo, con una breve descripción de la solución adoptada:

- Sistema de introducción de datos: Cuando la aplicación muestra la pantalla de introducir datos, el jugador tiene que indicar el valor de sus activos financieros. En las primeras versiones, la persona tenía que escribir a mano la cantidad exacta de cada producto, obligando a invertir un tiempo y rompiendo el ritmo de juego. Finalmente se decidió que cada jugador sólo debía indicar el número de cada tipo, siendo mucho más eficiente esta acción. Para ello se integró la librería externa *NumberPicker* (ya que el que viene por defecto en Android Studio es estéticamente muy mejorable).
- Representación gráfica de las puntuaciones finales: Debido al carácter financiero del juego de mesa, es deseable que los jugadores puedan ver un historial de sus puntuaciones en una partida finalizada, y la aportación en ésta de cada producto financiero. Para ello se implementó en la interfaz de usuario las gráficas requeridas, las cuales hemos obtenido con la librería *MPAndroidChart*.
- Creación de los servicios web: Sin duda la dificultad de mayor peso de todo el proyecto. Esto es debido que deben ser programados en el lenguaje PHP, el cual es completamente desconocido por el desarrollador. Pero el tiempo invertido en ellos no sólo ha sido la codificación, sino también en su depuración.
- Identificación del usuario y autorización: Como sabemos, para hacer uso de los servicios web, el cliente debe estar autorizado. Como hemos explicado, el uso de Tokens es idóneo para este propósito. Sin embargo, implementarlo en la aplicación no es inmediato. Finalmente, se decidió que el usuario recibiría una clave cifrada cada vez que iniciase sesión, la almacenaría en local de forma persistente, y la adjuntaría en las peticiones http mandadas al servidor. Allí, pasará la clave por un proceso interno para comprobar si corresponde a algún usuario.
- Búsqueda de un hosting para el servidor: La dificultad que más irritante. Inicialmente, tenían alojado los servicios web en un sitio web de hosting gratuitos. Sin embargo, la pesadez de tener que subir los scripts cada vez que han sido modificados, las continuas desconexiones, la dificultad de depuración, y la generación de errores desconocidos, nos obligó a buscar una alternativa. Esa opción fue utilizar el programa XAMPP, que permite crear un servidor de prueba en la red local (con su gestor de BBDD) y obtiene los scripts automáticamente de la carpeta indicada.
- Actualización dinámica de la información: La última dificultad a la que hacer frente, además una de las más didácticas. En el transcurso de una partida online, los jugadores deben ser conscientes en todo momento del estado de la partida y del resto de participantes, así como de los eventos que surjan. Para tratar con este reto, se barajó usar los mensajes push y FireBase. Sin embargo, la complejidad de implementar estas tecnologías (implicaría rehacer gran parte del sistema de servicios web), hizo declinarme por una metodología más simple, que es la escucha activa por parte del cliente (mandar peticiones periódicas al servidor), para respetar el trabajo realizado. Sin embargo, y como se indica en el siguiente apartado, la integración en el proyecto de FireBase es deseable.

8.2 Línea futura de desarrollo

Hay una serie de pasos claros que han quedado pendientes en el trabajo, que aumentarán el valor del producto y harán el sistema creado más sólido y profesional.

El primero es el de migrar los servicios web a un servidor de pago, que satisfaga los requisitos, para que cualquiera que se descargue la aplicación pueda acceder a las funciones online que ofrece. Sería buena idea crear un sitio web en Hostinguer (https://www.hostinger.es/), que ofrece una gran variedad de subscripciones según las necesidades de cada persona. Actualmente, los servicios web se encuentran alojador en un portal web con suscripción gratuita, con limitaciones importantes como no permitir peticiones HTTP de tipo UPDATE y DELETE, comprometiendo el correcto funcionamiento de secciones claves del sistema.

El segundo paso, y el que pondría a la vanguardia de las aplicaciones móviles distribuidas el programa, es agregar FireBase a nuestro proyecto. FireBase es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles distribuidas de Google. Ofrece una gran cantidad de servicios, como la posibilidad de ofrecer datos estadísticos sobre el uso de la aplicación, almacenamiento en la nube, hosting, y enviar mensajes orientados a los clientes (notificaciones push). Esta última característica sería la más interesante para nosotros, ya que la carga de trabajo en las partidas online se reduciría drásticamente en los clientes (actualmente mandan peticiones periódicamente preguntado por el estado) y el servidor (tiene que procesar y responderlas), de esta manera sería el servidor quien avisa a los consumidores sobre un evento, y éstos actuarían en consecuencia. Pero para utilizar esta ventaja, habría que modificar el sistema de identificación, autorización y registro por el impuesto por la plataforma nueva, e integrar también FireBase en los servicios web. Este paso requiere de mucho tiempo que no se ha dispuesto durante el transcurso del trabajo fin de carrera, pero se propone para una futura versión comercial del sistema.

El tercer paso sería terminar el módulo de *Partida Online*, ya que debido a la falta de tiempo (y la envergadura del proyecto), sólo se consiguió implementar las tres primeras pantallas (creación, inicialización y escenario online). Esta mejora está muy relacionada con lo comentado en el párrafo anterior, ya que la implementación de FireBase en el sistema hará posible el desarrollo de esta funcionalidad.

El cuarto paso corresponderá localizar la aplicación en distintos idiomas (originalmente sólo está en español), siendo el prioritario el inglés, seguido del francés y alemán.

Por último, sería deseable programar el software de manera que sea compatible con más formatos de pantalla y resolución, además de permitir la orientación horizontal de la pantalla (muy útil en dispositivos tablet).

8.3 Conclusión

La realización de este trabajo fin de grado ha sido una gran oportunidad para profundizar en los conceptos aprendidos en la carrera, relacionados con la rama de telemática. Es más, he obtenido nuevos conocimientos sobre tecnologías que me eran totalmente desconocidas. Soy consciente del rumbo que toma la sociedad, consumiendo cada vez más aplicaciones móviles, más complejas, atractivas visualmente e integradas en una comunidad online que permita a las personas interactuar con otras.

Sin embargo, la codificación del sistema implementado ha sido posible gracias al trabajo de ingeniería de software realizado. La especificación de requisitos me ha permitido mejorar mis habilidades para entender las exigencias del cliente y estructurar bien las necesidades y objetivos que debe cumplir un software. Además, ha sido la primera vez que he tenido que hacer el diseño de una aplicación Android, resultando un proceso muy pedagógico.

En conclusión, este proyecto me ha brindado experiencia en el desarrollo de aplicaciones Android distribuidas que considero muy valiosa, he profundizado en conceptos vistos en el grado como las tareas del desarrollo software, y me ha convertido en una persona más competente en el ámbito del desarrollo de aplicaciones móviles distribuidas.

REFERENCIAS BIBLIOGRÁFICAS

[1] Isabel Roman Martinez, Apuntes de la asignatura Ingeniería de Software,
Departamento de Ingeniería Telemática, Universidad de Sevilla.
[2] Jose Manuel Fornés Rumbao, Apuntes de la asignatura Diseño de aplicaciones móviles,
Departamento de Ingeniería Telemática, Universidad de Sevilla.
[3] Roberto Montero Miguel, 2014, Guía Práctica de Java 8, Madrid, Anaya Multimedia.
[4] Jesús Tomás, 2017, El gran libro de Android, Tercera edición, Barcelona, Marcombo.
[5] Wikipedia. Recuperado 15 de junio2018, desde
https://es.wikipedia.org/wiki/MagicDraw_UML
[6] WebHost. Recuperado el 15 de junio de 2018, desde
https://www.000webhost.com/
[7] Hermosa Programación. Recuperado el 15 de junio de 2018, desde
http://www.hermosaprogramacion.com/2015/10/servicio-web-restful-android-php-mysql-json/
[8] Codictados. Recuperado el 15 de junio de 2018, desde
http://codictados.com/volley-web-services-para-principiantes/
[9] Hermosa Programación. Recuperado el 15 de junio de 2018, desde
http://www.hermosaprogramacion.com/2015/02/android-volley-peticiones-http/
[10] Carlos Azaustre. Recuperado el 15 de junio de 2018, desde
https://carlosazaustre.com/que-es-la-autenticacion-con-token/
[11] Android Developers. Recuperado el 15 de junio de 2018, desde
https://developer.android.com/studio/publish/?hl=es-419
[12] Android Developers. Recuperado el 15 de junio de 2018, desde
https://developer.android.com/guide/components/fragments?hl=es-419
[13] Android Developers. Recuperado el 15 de junio de 2018, desde
https://developer.android.com/reference/android/app/Activity
[14] Junta de Andalucía, MADEJA. Recuperado el 15 de junio de 2018, desde
http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/407
[15] Hermosa Programación. Recuperado el 15 de junio de 2018, desde
http://www.hermosaprogramacion.com/2016/06/android-push-notifications-firebase-cloud-messaging/
[16]Firebase. Recuperado el 15 de junio de 2018, desde
http://www.hermosaprogramacion.com/2016/06/android-push-notifications-firebase-cloud-messaging/
[17] travijuu/NumberPicker. GitHub. Recuperado el 15 de junio de 2018, desde
https://github.com/travijuu/NumberPicker
[18] Google/Volley. GitHub. Recuperado el 15 de junio de 2018, desde
https://github.com/google/volley

[19] PhilJay/MPAndroidChart. GitHub. Recuperado el 15 de junio de 2018, desde
https://github.com/PhilJay/MPAndroidChart
[20] barteksc/AndroidPdfViewer. GitHub. Recuperado el 15 de junio de 2018, desde
https://github.com/barteksc/AndroidPdfViewer
[21] Android Studio. Recuperado el 15 de junio de 2018, desde
https://developer.android.com/studio/
[22] Wikipedia. Recuperado el 15 de junio de 2018, desde
https://es.wikipedia.org/wiki/JSON

GLOSARIO

IDE: Entorno de Desarrollo Integrado

BBDD: Base de Datos

API: Application Programming Interface

URL: Uniform Resource Locator

CASE: Computer Aided Software Engineering

REST: Representational State Transfer

SOAP: Simple Object Access Protocol

UML: Unified Modeling Language

SDK: Software Development Kit

GUI: Graphical User Interface